Spring 2016

# Common Techniques in Graceful Tree Labeling with a New Computational Approach

Michael Guyer

# COMMON TECHNIQUES IN GRACEFUL TREE LABELING WITH A NEW COMPUTATIONAL APPROACH

A Thesis

Submitted to the McAnulty College & Graduate School of Liberal Arts

Duquesne University

In partial fulfillment of the requirements for

the degree of Master of Science

By

Michael Guyer

May 2016

COMMON TECHNIQUES IN GRACEFUL TREE LABELING WITH A NEW COMPUTATIONAL APPROACH

By

Michael Guyer

Approved April 1, 2016

---

Dr. Karl Wimmer
Associate Professor of Mathematics
(Committee Chair)

---

Dr. John Kern
Associate Professor of Statistics
(Department Chair)

---

Dr. Rachael Neilan
Assistant Professor of Mathematics
(Committee Member)

---

Dr. James Swindal
Dean, McAnulty College and Graduate School of Liberal Arts
Professor of Philosophy

ABSTRACT

COMMON TECHNIQUES IN GRACEFUL TREE LABELING WITH A NEW COMPUTATIONAL APPROACH

By

Michael Guyer

May 2016

Thesis supervised by Dr. Karl Wimmer.

The graceful tree conjecture was first introduced over 50 years ago, and to this day it remains largely unresolved. Ideas for how to label arbitrary trees have been sparse, and so most work in this area focuses on demonstrating that particular classes of trees are graceful. In my research, I continue this effort and establish the gracefulness of some new tree types using previously developed techniques for constructing graceful trees. Meanwhile, little work has been done on developing computational methods for obtaining graceful labelings, as direct approaches are computationally infeasible for even moderately large trees. With this in mind, I have designed a new computational approach for constructing a graceful labeling for trees with sufficiently many leaves. This approach leverages information about the local structures present in a given tree in order to construct a suitable labeling. It has been shown to work for many small cases and thoughts on how to extend this approach for larger trees are put forth.

ACKNOWLEDGEMENT

I wish to thank my adviser Dr. Karl Wimmer for introducing me to the "disease" that is the graceful tree conjecture and for his continued guidance and support throughout my research efforts. He has been a tremendous source of knowledge both as an instructor and an adviser and my experience as a graduate student at Duquesne would undoubtedly have suffered without his influence. I also want to extend my gratitude to Dr. Rachael Neilan who has provided additional feedback as my thesis reader and has been a pleasure to have as an instructor. I would be remiss to not acknowledge as well Dr. John Kern's consistent enthusiasm and interest in my efforts as a student.

Additional thanks must be given to my fellow graduate students and officemates Adam Baumgardner and Rich Boyles. They have been a steady network of support and I cannot thank them enough for the hours of shared frustration, excitement, and laughs that helped carry me to the finish line and allow me to complete this thesis. I cannot imagine my time in this program without their friendship.

Finally, I wish to thank my family, which includes my mother Dottie, my father Michael, my brother Justin, and my sister Jenna. They have been my backbone through difficult times and personal struggles in recent years, and it is no exaggeration to suggest that I likely would not have finished this program without their emotional support. The same thanks must be offered to my close friend and roommate Ed Witt as well as my girlfriend Rachael Pennock, who have lent their ears to my concerns during many late night conversations.

Page

# Contents

LIST OF FIGURES

Page

# 1 Introduction

## 1.1 Definitions

We begin with some basic definitions.

**Definition 1.1.** A *graph* $G$ is defined by a vertex set $V(G)$ and an edge set $E(G) \subset V(G) \times V(G)$, i.e. $G = (V(G), E(G))$. For simplicity, throughout this paper we will denote an edge $(u, v)$ by $uv$.

**Definition 1.2.** For a given graph $G$, we define a *labeling* $f$ of $G$ to be an injective mapping from the vertex set $V(G)$ to the non-negative integers. The *label* of a vertex $v \in V(G)$ is denoted by $f(v)$, and the *induced label* of an edge $uv \in E(G)$ is given by the absolute difference of the labels of the vertices it adjoins, i.e. $|f(u) - f(v)|$.

Of particular interest in this text are *$\beta$-labelings*, otherwise known as *graceful labelings*. We will also at times consider *$\alpha$-labelings*. We define these labelings below.

**Definition 1.3.** Let $f$ be a labeling with vertex labels $V_f$ and induced edge labels $E_f$. $f$ is a *$\beta$-labeling*, or *graceful labeling*, if $V_f \subset \{0, 1, ..., n\}$ and $E_f = \{1, ..., n\}$. A graph $G$ that has a graceful labeling $f$ is said to be *graceful*.

**Definition 1.4.** A labeling $f$ of a graph $G$ is an *$\alpha$-labeling* with *critical number* $k$ if $f$ is a graceful labeling and $k$ is an integer such that for each edge $uv$ of $G$, either $f(u) \leq k < f(v)$ or $f(v) \leq k < f(u)$ holds.

We have defined labelings and the notion of gracefulness for general graphs. Throughout this paper, we will be restricting our focus specifically to trees, which we define below.

**Definition 1.5.** A graph $G$ is said to be *connected* if for every pair of vertices $u, v \in V(G)$ there is a sequence of edges in $E(G)$ extending from $u$ to $v$, i.e. a sequence of edges of the form $ux_1,\ x_1x_2,\ x_2x_3,\ ...,\ x_{n-1}x_n,\ x_nv$ (otherwise known as a path, which we define in Section 2).

**Definition 1.6.** A (connected) graph $T$ is a *tree* if it is minimally connected, i.e. removing any edge will result in the graph no longer being connected.

**Remark:** We state now without proof the following fact about trees: A tree with $n$ vertices has $n-1$ edges. Thus, a graceful labeling $f$ of a tree T with $n$ vertices has vertex labels $V_f = \{0, 1, ..., n-1\}$ and induced edge labels $E_f = \{1, ..., n-1\}$.

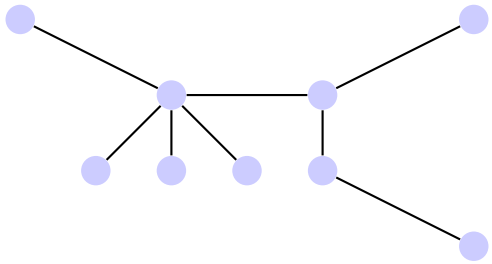

Figure 1.1: An example of a tree with 9 vertices. Note that this tree has $9 - 1 = 8$ edges.

## 1.2 Graceful Tree Conjecture

**Definition 1.7.** Let $G$ be a graph. A *decomposition* of $G$ is a set of subgraphs $H_1, H_2, ..., H_k$ such that each edge in $E(G)$ belongs to exactly one $H_i$, i.e. it is a partition of the edges of $G$.

**Definition 1.8.** Let $G$ be a graph with $n$ vertices and $v_i v_j$ be an edge of $G$. We say that increasing both indices by one to yield the edge $v_{i+1}v_{j+1}$, where indices are taken modulo $n$, is *turning* the edge $v_i v_j$. *Turning* a subgraph $H$ of $G$ is the process of turning every edge of $G$ simultaneously.

**Definition 1.9.** A decomposition $H_1, H_2, ..., H_k$ of a graph $G$ is said to be a *cyclic decomposition* if for each $i$, turning $H_i$ yields $H_j$ for some $j \neq i$.

We are now in a position to introduce the Ringel-Kotzig conjecture, otherwise known as the Graceful Tree Conjecture. Motivated by previous conjectures suggesting that the complete graph $K_{2n+1}$ (i.e. the graph with $2n+1$ vertices where every pair of vertices is

connected by an edge) can be (cyclically) decomposed into $2n+1$ subgraphs isomorphic to a given tree $T$ with $n$ edges, the Graceful Tree Conjecture claims that all trees are graceful. It has been shown that the truth of the these earlier conjectures related to graph decompositions would follow from confirmation of the Graceful Tree Conjecture [19]. This conjecture has remained an open problem for over 50 years, but progress has been made for certain classes of trees. We introduce some of these results in Section 2, but first we briefly discuss some applications of graceful labelings.

## 1.3 Applications

Graceful tree labeling has applications to other combinatorial problems within discrete mathematics, but it also has many practical applications. Such applications include models in coding theory, ambiguities in x-ray crystallography, circuit design, communication network addressing, and database management, among others [7]. This represents a wide variety of applications, which reflects the versatility of graphs, and trees in particular, to model real world phenomena.

# 2 Some Basic Results

**Note:** All proofs presented in this section are my own unless otherwise stated.

## 2.1 Paths

**Definition 2.1.** A (finite) *path* is a graph of the form $V(G) = \{v_1, v_2, ..., v_k\}, E(G) = \{v_1v_2, v_2v_3, ..., v_{k-1}v_k\}$, and we denote the path containing $n$ edges (and hence $n+1$ vertices) by $P_n$. We note that $P_n$ is a tree for every $n \in \mathbb{N}$.

**Theorem 2.2** (Rosa, [19])**.** *$P_n$ is graceful for every $n$.*

*Proof:* Starting from either end point, we alternatively label vertices with the lowest or highest number from $\{0, 1, ..., n\}$ still available as we traverse the path. This yields a labeling of $0, n, 1, n-1, 2, n-2, ...$ where the induced labels of the edges are, in order, $n, n-1, n-2, ..., 1$. □

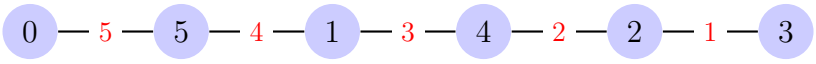


Figure 2.1: A graceful labeling of $P_5$ with edge labels included.

## 2.2 Caterpillars

We first define the general notion of an *m-distant* tree.

**Definition 2.3** (Morgan, [17])**.** A tree $T$ is *m-distant* if there exists a path $P$ in $T$ such that any vertex of $T$ is at most a distance of $m$ from $P$.

We are now in a position to define "caterpillar" graphs.

**Definition 2.4.** A *caterpillar* is a 1-distant tree.

In a sense, caterpillars can be viewed as one level of complexity above paths. We might hope that the gracefulness of caterpillars can be as readily established as it was for paths.

**Theorem 2.5** (Rosa, [19]). *All caterpillars are graceful.*

*Proof:* Let $C$ be a caterpillar with $n$ edges and let $P = P_k$ be its central path. As previously done for paths, begin at either end point (say $v_1$) of $P$ and assign it the lowest available label of 0. Moving to $v_2$, the next vertex in $P$, we assign it the highest available label of $n$. We then consider the unlabeled neighbor(s) of $v_2$, including $v_3$, the next vertex in $P$. If $v_2$ has $l$ unlabeled neighbors, we assign these neighbors the $l$ lowest available labels (in this case $1, 2, ..., l$), where $v_3$ gets the highest of these labels, i.e. $l$. We then consider the neighbors of $v_3$ that have not yet been labeled (i.e. excluding $v_2$). If $v_3$ has $j$ unlabeled neighbors, we assign these neighbors the $j$ highest available labels (in this case $n-1, n-2, ..., n-j$), where $v_4$ gets the lowest of these labels, i.e. $n-j$. We continue in a similar fashion, where at each $v_i$ in $P$ we assign its $m_i$ unlabeled neighbors with either the $m_i$ highest available labels or the $m_i$ lowest available labels, depending on whether $v_i$ was labeled with a "low" value or a "high" value, respectively. In each case, $v_{i+1}$ is labeled with either the highest or the lowest of these values. We proceed in this manner until reaching $v_{k+1}$, the end of our path, at which point we have yielded a graceful labeling. □

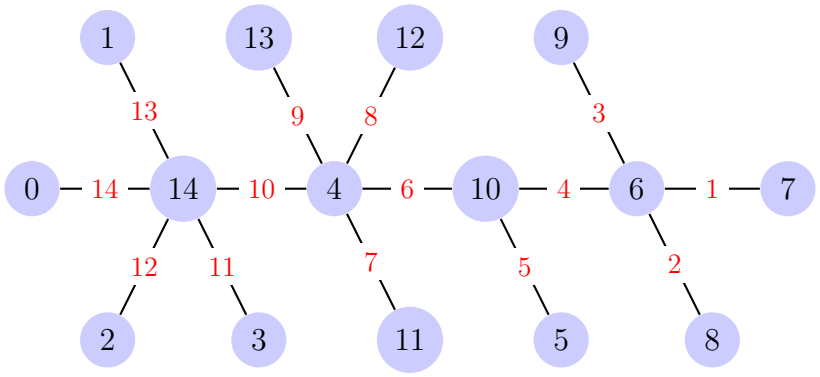


Figure 2.2: An example of a caterpillar graph and the graceful labeling yielded from the approach described in the above proof.

**Definition 2.6.** Let $G$ be a graph with $n$ edges, and let $f$ be a labeling of $G$. The labeling $f'$ defined by $f'(v) = n - f(v)$ is called the *complementary labeling* of $f$.

**Lemma 2.7.** *Let $G$ be a graph with $n$ edges, and let $f$ be a graceful labeling of $G$. Then the complementary labeling $f'$ of $G$ is also graceful.*

*Proof:* We have that $f$ is injective with range $\{0, 1, ..., n\}$, and so it holds that $f'$ is also injective with range $\{0, 1, ..., n\}$. Furthermore, we have that for any edge $uv$ of $G$, $|f(u) - f(v)| = |(n - f(u)) - (n - f(v))| = |f'(u) - f'(v)|$, and so the induced labels of the edges are the same as well. □

**Remark:** From the above lemma, we see that in our previous proofs establishing the gracefulness of paths and caterpillars, we could have just as easily begun our labelings with the highest available vertex label, then labeled its neighbor with the lowest available vertex label, then labeled its neighbor(s) with the highest available vertex label(s), etc., i.e. using the complementary labelings of those provided above.

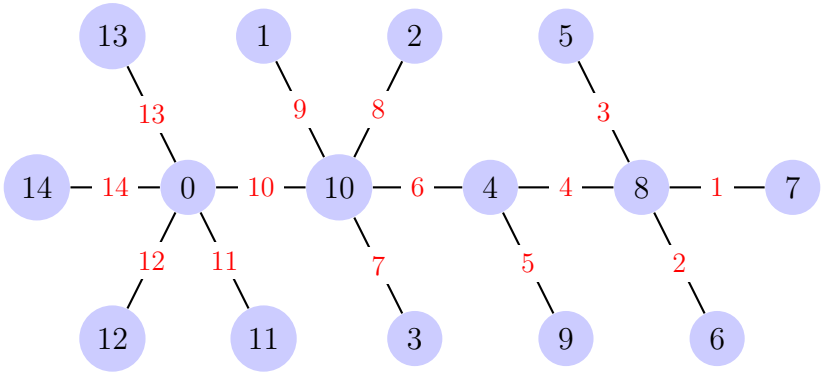


Figure 2.3: The caterpillar from Figure 2.2 labeled with the complementary labeling. Note that the induced edge labels are the same and only the vertex labels have changed.

At times in this paper we will consider a specific type of caterpillar graph, which we define below.

**Definition 2.8.** Let $T$ be a tree consisting of a single central vertex $v_0$ and where all other vertices of $T$ are adjacent only to $v_0$. We say that $T$ is a *star*, and we denote the star where $v_0$ has $k$ neighbors by $S_k$.

The star $S_k$ can be labeled gracefully in the same manner as any other caterpillar, but the simplest way to label it is to assign a label of 0 to the central vertex $v_0$ and the labels $1, 2, ..., k$ to its $k$ neighbors. When considering star graphs at later points in this paper, we shall label it in this manner.

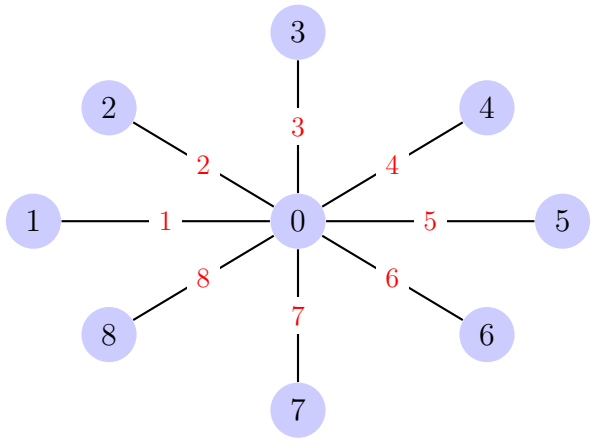


Figure 2.4: The star $S_8$ with the graceful labeling described above.

## 2.3 Lobsters

**Definition 2.9.** A *lobster* is a 2-distant tree.

In the previous sections, we first defined paths and quickly established their gracefulness. We then defined caterpillar graphs and proved they are graceful using a modified version of the approach used for paths. It would be reasonable to assume that lobsters are graceful as well and that this can be shown in a similar fashion, but this is in fact not the case. It is still not known whether all lobsters are graceful, although it was conjectured by Bermond in 1979 [3]. However, the gracefulness of some classes of lobsters has been established. The most notable result is the following, which is due to Morgan and will be stated without proof.

**Definition 2.10.** A *matching* of a graph $G$ is a subset $M \subset E(G)$ such that every vertex of $G$ is an endpoint of at most one edge in $M$. A *perfect matching* of $G$ is a matching such that every vertex of $G$ is an endpoint of exactly one edge in the matching.

**Theorem 2.11** (Morgan, [17])**.** *All lobsters that contain a perfect matching are graceful.*

## 2.4 Spiders

**Definition 2.12.** The *degree* of a vertex $v$ in a graph $G$ is given by the number of vertices that $v$ is adjacent to in $G$, or equivalently the number of edges that $v$ is an endpoint of in $G$. We say that a vertex in $G$ of degree 1 is a *leaf* of $G$.

**Definition 2.13** (Bahls et al., [2])**.** Let $T$ be a tree with at most one vertex of degree greater than 2. We say that $T$ is a *spider tree*, and if such a vertex exists, it is said to be the *branch point* of the tree. A *leg* of a spider tree is a path from the branch point to a leaf of the tree.

An example of a spider tree is included below.

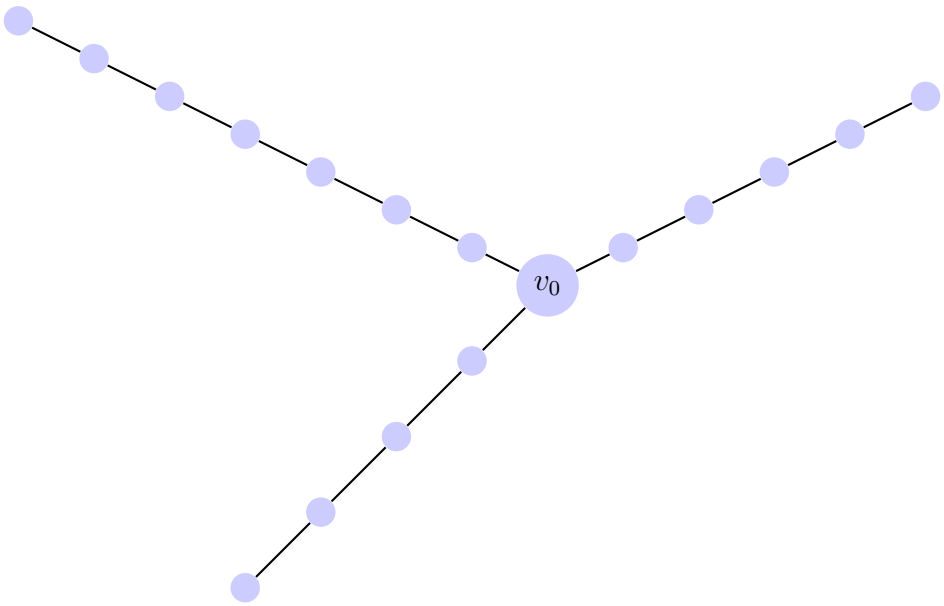


Figure 2.5: An example of a spider tree with branch point $v_0$.

We now briefly state some of the primary results related to the gracefulness of spider trees.

**Theorem 2.14** (Bahls et al., [2])**.** *Let $T$ be a spider tree where each leg has length $m$ or $m+1$ for some $m \geq 1$. Then $T$ is graceful.*

**Theorem 2.15** (Huang et al., [15])**.** *Let $T$ be a spider tree with three legs of any length, i.e. $T$ has three leaves. Then $T$ is graceful.*

## 2.5 Symmetrical Trees

**Definition 2.16.** Let $T$ be a rooted tree in which every level contains vertices of the same degree. Then we say that $T$ is a *symmetrical tree.*

Below is an example of a symmetrical tree.

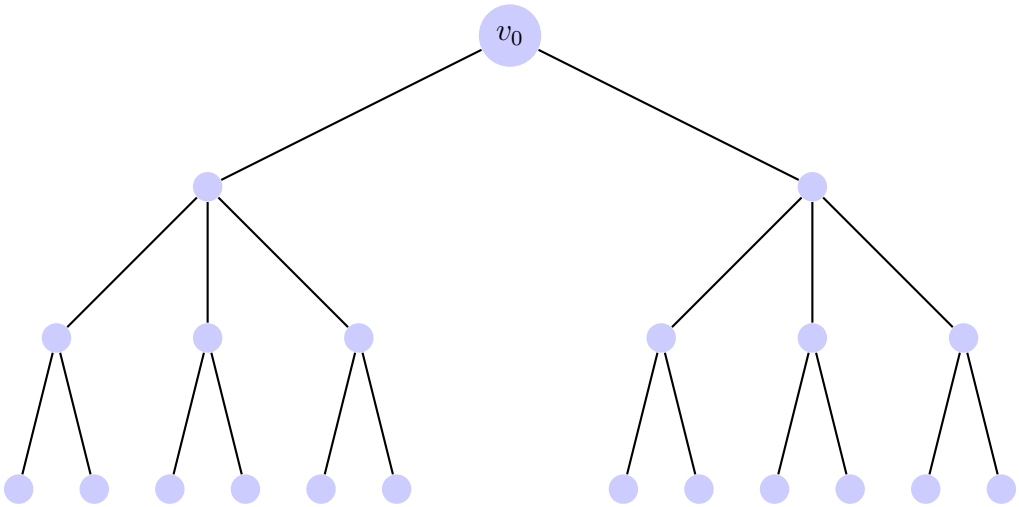


Figure 2.6: A symmetrical tree $T$ with root $v_0$.

**Theorem 2.17** (Bermond and Sotteau, [4])**.** *All symmetrical trees are graceful.*

*Proof:* We paraphrase this argument from a survey of Robeva [18]. The proof is established using induction on the number of levels. We will exclude the technicalities of the proof but will summarize the basic idea behind the inductive step. We consider a rooted symmetrical tree $T$ with $k$ children (neighbors) $v_1, v_2, ..., v_k$ (ordered from left to right) and note that each of its $k$ subtrees $T_1, T_2, ..., T_k$ (where $T_i$ is the subtree with $v_i$ as the root) are graceful and isomorphic. We begin by assigning each of these isomorphic subtrees with the same graceful labeling and then increase the values of some of their vertex labels in the following manner. Suppose that each of the $k$ subtrees has $n$ vertices. Beginning at the 0th level (the root) of each subtree and moving left to right, we add $(k-1)n$ to $v_1$, the root of $T_1$, $(k-2)n$ to $v_2$, the root of $T_2$, etc., i.e. add $(k-i)n$ to $v_i$, the root of $T_i$ for each $i \in \{1, ..., k\}$. We now move to the 1st level of each subtree and moving right to left, we add $(k-1)n$ to each of the vertices

in the 1st level of $T_k$, $(k-2)n$ to each of the vertices in the 1st level of $T_{k-1}$, etc., i.e. add $(k-i)n$ to each of the vertices in the 1st level of $T_{k-i+1}$ for each $i \in \{1, ..., k\}$. Continuing this pattern of alternating direction, we move to the 2nd level of each subtree and moving left to right, we add $(k-1)n$ to each of the vertices in the 2nd level of $T_1$, $(k-2)n$ to each of the vertices in the 2nd level of $T_2$, etc., i.e. add $(k-i)n$ to each of the vertices in the 2nd level of $T_i$ for each $i \in \{1, ..., k\}$. This pattern of alternating direction repeats until finishing with the last level in each subtree. Finally, we assign $nk+1$, the maximum label, to the root of $T$, and then take the complimentary labeling of this labeling that we have constructed. This complimentary labeling is what can be used to inductively label symmetrical trees with a greater number of levels. $\square$

**Remark:** While we take the complementary labeling in the above proof as part of the induction, we note that the above shows that all symmetrical trees have a graceful labeling where the root is given the maximum label. This is a fact that we will utilize heavily in Section 4.

We now demonstrate how this inductive approach can be used to label the symmetrical tree $T$ in Figure 2.6.

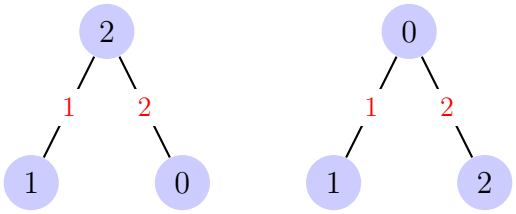


Figure 2.7: The graceful labeling first obtained for the smallest nontrivial subtree of $T$ followed by the complementary labeling used for the next inductive step.

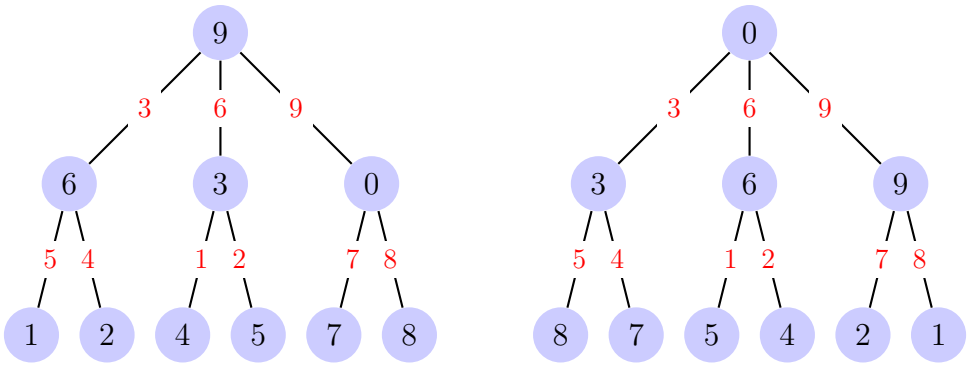


Figure 2.8: The graceful labeling first obtained of the isomorphic subtree rooted at each child of $v_0$ in $T$, followed by the complementary labeling used for the next inductive step.

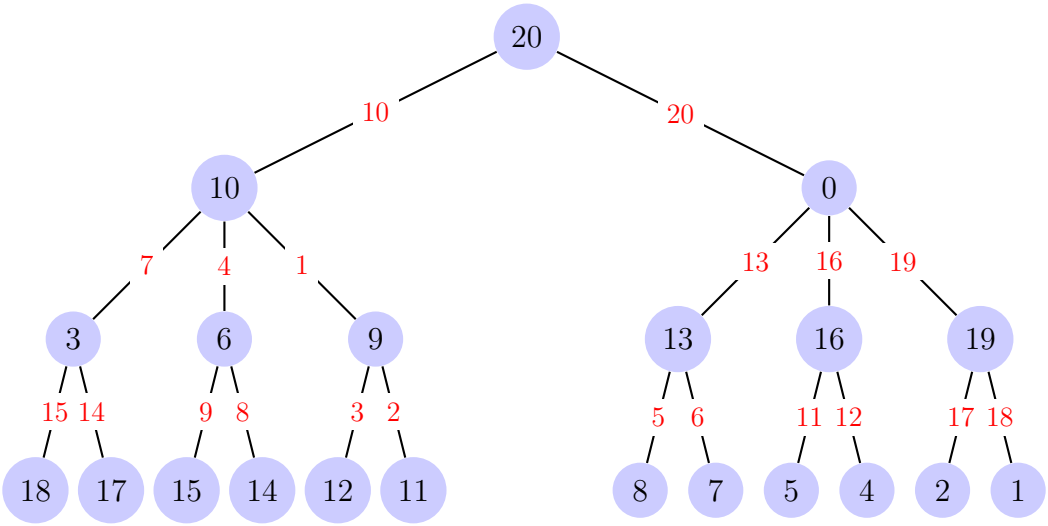


Figure 2.9: The graceful labeling first obtained of $T$ where the root has the maximum label.

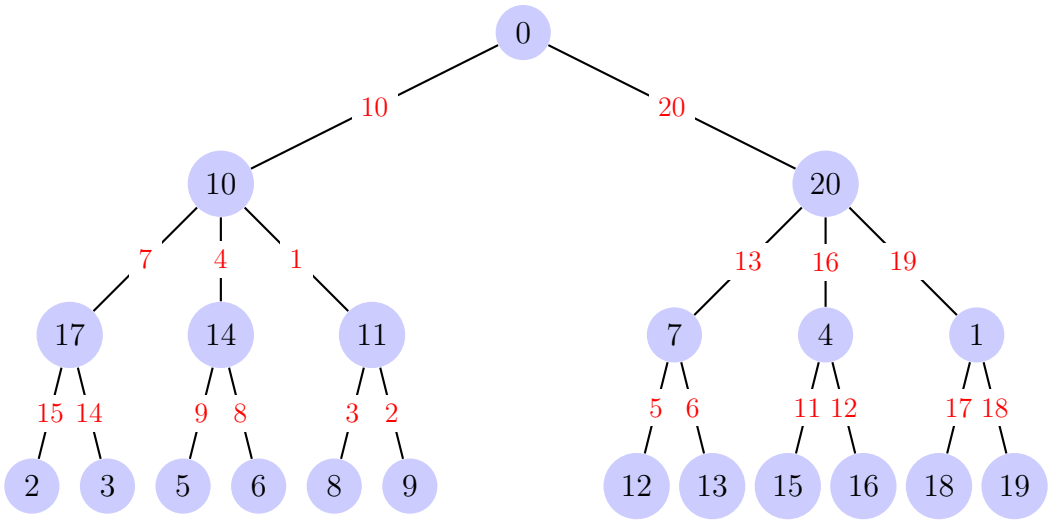


Figure 2.10: The final graceful labeling of the symmetrical tree $T$.

## 2.6 Trees of Sufficiently Small Size

Several results exist that demonstrate gracefulness of trees that are "sufficiently small" in some sense. In order to discuss some of these results, we must define some new terms below.

**Definition 2.18.** The *eccentricity* of a vertex $v$ in a graph $G$ is the maximum distance between $v$ and any other vertex in $G$, where distance is defined by the number of edges in a shortest path between $v$ and some other vertex.

**Definition 2.19.** The *diameter* of a graph $G$ is given by the maximum eccentricity of any vertex in the graph.

**Theorem 2.20** (Hrnčiar and Haviar, [14])**.** *All trees of diameter at most five are graceful.*

The proof is due to Hrnčiar and Haviar [14], using a transfer technique that we shall discuss in much more detail in the next section on existing techniques of interest.

Along the way to proving the gracefulness of diameter five trees, Hrnčiar and Haviar prove the gracefulness of diameter four trees with use of the following lemma:

**Lemma 2.21** (Hrnčiar and Haviar, [14])**.** *Every tree $T$ of diameter four with a central vertex of odd degree has a graceful labeling such that the label of the central vertex is maximal.*

In Section 4 we will consider the veracity of an equivalent statement for diameter four trees with a central vertex of even degree and discuss the implications.

Some results have been established on the gracefulness of trees that have sufficiently few vertices. These results were confirmed with the use of computer algorithms, and such algorithms and computational approaches will be elaborated on in further sections and are summarized below.

**Theorem 2.22** (Aldred and McKay, [1]). *All trees with at most* 27 *vertices are graceful.*

This above result was found by Aldred and McKay in 1998 with the use of a hill-climbing computer algorithm. It was improved in 2010 by a project led by Wenjie Fang.

**Theorem 2.23** (Fang, [10]). *All trees with at most* 35 *vertices are graceful.*

These results lend credibility to the Graceful Tree Conjecture in that trees with at most 35 vertices represent an incredible number of trees with a variety of structures, suggesting that the search for the proof of this conjecture, while elusive, is worthwhile.

# 3 Existing Techniques of Interest

## 3.1 Joining

Most of the ideas and results discussed in this section are due to Shamik Ghosh [12]. They serve as a foundation for some of my own research efforts to be discussed in Section 4.

**Definition 3.1.** Let $G = (V(G), E(G))$ be a graceful graph with $n$ vertices, $m$ edges, and a graceful labeling $f$. Suppose $V(G) \subseteq \{v_0, v_1, ..., v_m\}$, where $f(v_i) = i$ for all $v_i \in V(G)$. Let $G_1 = (V(G_1), E(G_1))$ be an isomorphic copy of $G$ with a graceful labeling $f_1$ such that $V(G_1) = \{u_i \mid v_i \in V(G)\}$ and $f_1(u_i) = i = f(v_i)$ for all $u_i \in V(G_1)$. Fix some $j \in \{i \mid v_i \in V(G)\}$ and let $G_2$ be the graph obtained from $G$ and $G_1$ by joining the vertices $v_j$ and $u_j$ with an edge. Then $G_2$ is called a *double of $G$ at $j$*.

**Remark:** In the above definition, if $G$ is a graceful tree, then $n = m + 1$, and thus $V(G) = \{v_0, v_1, ..., v_m\}$.

**Proposition 3.2** (Ghosh, [12])**.** *Any double of a completely graceful graph with $m$ edges has a complete $\alpha$-labeling with a critical number $m$.*

The proofs of this result and other results in this section use an adjacency matrix technique to establish the existence of an $\alpha$-labeling (or simply graceful labeling in some cases) and provide an explicit labeling. The ideas behind this approach are discussed below.

**Definition 3.3.** Let $A$ be an $m \times n$ matrix. The *box-value* of the position $(i, j)$ of $A$ is $m + j - i$. A *diagonal* of $A$ is a set of positions of $A$ with the same box-value, i.e. for each $c = 1, 2, ..., m + n - 1$, the set $d_c = \{(i, j) \mid m + j - i = c\}$ is a diagonal. An $m \times n$ binary matrix $A$ (i.e., a matrix where all entries are either 0 or 1) is said to be *graceful* if every diagonal of $A$ contains at most one 1 and $A$ is said to be *completely graceful* if every diagonal of $A$ contains exactly one 1 (except the principal diagonal of an adjacency matrix of a graph where all the entries are 0).

**Theorem 3.4** (Bloom, [6]). *Let $G$ be a graph with $n$ vertices and $m$ edges such that $n = m+1$. Then $G$ is completely graceful if and only if the vertices of $G$ can be arranged in such a way that its adjacency matrix becomes completely graceful.*

*Proof:* The basic idea behind the proof is as follows. First, suppose $G$ is completely graceful. We may arrange the adjacency matrix of $G$ in such a way that the rows and columns are in ascending order with regards to the labels of the vertices (this is known as a *canonical adjacency matrix*). Each diagonal of this matrix has exactly one 1, with the exception of the principal diagonal which has all 0 entries, and so it is a completely graceful matrix.

On the other hand, if we instead suppose that we may arrange the vertices of $G$ in such a way that its adjacency matrix becomes completely graceful, then this yields a graceful labeling of $G$. To see this, recall that each diagonal of a square matrix corresponds to a set of positions that all have the same box-value. For box-values of $c = 1, 2, ..., m-1$, i.e. diagonals below the principal diagonal, entries in those diagonals can be seen to correspond with differences of $m - c$, i.e. $m, m-1, ..., 1$ respectively between two vertex labels $i$ and $j$ given by the position $(i, j)$ of an entry in the matrix. Since each diagonal has exactly one 1, this corresponds to a labeling where a bijection exists between the induced edge labels and the set $\{1, 2, ..., m\}$. For box-values of $c = m+1, m+2, ..., m+n-1$ (above the principal diagonal), where $m + n - 1 = m + (m + 1) - 1 = 2m$, entries in these diagonals correspond to edge labels of $c - m$ for each $c = m+1, m+2, ..., 2m$, i.e. these entries correspond to the exact same set of induced edge labels. Indeed, the diagonals above the principal diagonal are simply a transpose of those below the principal diagonal, and so represent the exact same vertex and induced edge labels that yield a graceful labeling of $G$. □

**Remark:** The use of the separate terms graceful and completely graceful in the above definition and theorem distinguishes between graphs where $n \leq m + 1$ and so $V(G) \subseteq \{v_0, v_1, ..., v_m\}$ and graphs where $n = m + 1$ and thus $V(G) = \{v_0, v_1, ..., v_m\}$, where as usual $n$ denotes the number of vertices and $m$ denotes the number of edges. We note that if $n > m + 1$ for a graph $G$ then it cannot have a graceful labeling, as no injective mapping exists between $V(G)$ and $\{0, 1, ..., m\}$. The majority of our discussion pertains to trees,

where it is always the case that $n = m + 1$, and so there is no real distinction between the terms graceful and completely graceful and hence our strict use of the term graceful in both earlier and later sections is unambigious.

The relationship between completely graceful adjacency matrices and completely graceful graphs outlined above by Theorem 3.4 might perhaps be made clearer by an example. Consider the following graceful tree $T$ and its canonical adjacency matrix.

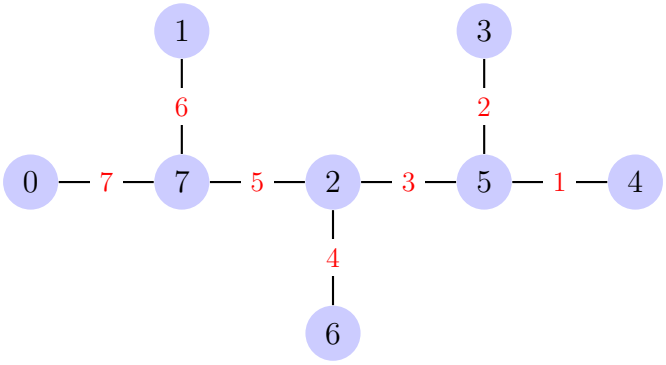


Figure 3.1: A graceful tree T.

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Figure 3.2: The canonical adjacency matrix of T.

We see in the above canonical adjacency matrix that other than the principal diagonal, each diagonal (moving downward and to the right) has exactly one 1. Let us take a closer look at this example. It turns out that the given labeling for $T$ is actually an $\alpha$-labeling with critical number 4. It is known that a graph with an $\alpha$-labeling is bipartite (Kotzig [16]), which we shall define here.

**Definition 3.5.** A graph $G = (V(G), E(G))$ is *bipartite* if $V(G)$ can be partitioned into disjoint sets $V_A(G), V_B(G)$ such that for any edge $uv \in E(G)$, either $u \in V_A(G)$ and $v \in V_B(G)$ or $u \in V_B(G)$ and $v \in V_A(G)$. In other words, all edges in $G$ connect a vertex from $V_A(G)$ to a vertex in $V_B(G)$.

With the above definition in mind, it is easy to see that any graph $G$ with an $\alpha$-labeling $f$ with critical number $k$ is bipartite. In fact, the critical number $k$ suggests a natural bipartition; $V_A(G) = \{v \in V(G) \mid f(v) \leq k\},\ V_B(G) = \{v \in V(G) \mid f(v) > k\}$.

**Definition 3.6.** Given a bipartite graph $G = (V_A(G), V_B(G), E(G))$, the submatrix $A$ of the adjacency matrix of $G$ containing rows corresponding to the vertices of $V_A(G)$ and columns corresponding to the vertices of $V_B(G)$ is called the *biadjacency matrix* of $G$.

The adjacency matrix of a bipartite graph $G = (V_A(G), V_B(G), E(G))$ is as shown below.

$$\begin{array}{c} \\ V_A(G) \\ V_B(G) \end{array} \begin{array}{c} \begin{array}{cc} V_A(G) & V_B(G) \end{array} \\ \begin{pmatrix} \mathbf{0} & A \\ A^T & \mathbf{0} \end{pmatrix} \end{array}$$

Figure 3.3: The adjacency matrix of a bipartite graph $G$ with a biadjacency matrix $A$.

We now return to our previous example of the graceful tree $T$ and its $\alpha$-labeling $f$ with a critical number of 4. We have that a bipartition is given by $V_A(T) = \{0, 1, 2, 3, 4\},\ V_B(T) = \{5, 6, 7\}$. We note now that the adjacency matrix of $T$ is of the form shown above for bipartite graphs, where the biadjacency matrix $A$ of $T$ is given by

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc} 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \end{array}$$

This discussion of bipartite graphs and their corresponding biadjacency matrices now allows us to state the following theorem, which will be stated without proof.

**Theorem 3.7** (Ghosh, [12]). *Let $G$ be a graph with $n$ vertices and $m$ edges. Then $G$ has a complete $\alpha$-labeling with a critical number $k$ if and only if $n = m + 1$, $G$ is bipartite and there is a bipartition of $G$ where the vertices of $G$ can be arranged in such a way that its biadjacency matrix $A$ becomes completely graceful and $A$ has $k + 1$ rows.*

**Remark:** We note that if a tree $T$ is bipartite with bipartition $(V_A(T), V_B(T))$ and hence has an $\alpha$-labeling with critical number $k$, we must have that either $k = |V_A(T)| - 1$ or $k = |V_B(T)| - 1$. To see this, we observe that we must either assign the vertices in $V_A(T)$ with the $|V_A(T)|$ smallest labels, which yields an $\alpha$-labeling with critical number $|V_A(T)| - 1$, or we must instead assign the vertices in $V_B(T)$ with the $|V_B(T)|$ smallest labels, which in turn yields an $\alpha$-labeling with critical number $|V_B(T)| - 1$. We further note that these two labelings are complements of one another.

We now return to the proof of Proposition 3.2.

*Proof of Proposition 3.2:* The following proof is given by Ghosh [12]. Let $G = (V(G), E(G))$ be a completely graceful graph with $m$ edges and a graceful labeling $f$. It follows from Theorem 3.4 that a canonical adjacency matrix $A$ of $G$ is completely graceful. Suppose $G_2$ is obtained from $G$ as in Definition 3.1. Let $A_1$ be the matrix obtained from A by adding a

1 in the position $(j, j)$. Let

$$A_2 = \begin{pmatrix} \mathbf{0} & A_1 \\ A_1^T & \mathbf{0} \end{pmatrix}$$

It is clear that $A_2$ is the adjacency matrix of $G_2$ and $A_1$ is the biadjacency matrix of $G_2$. Furthermore, $A$ contains only 0's along its principal diagonal, and so $A_1$ contains exactly one 1 along its principal diagonal. Therefore $A_1$ is completely graceful, and so by Theorem 3.7 $G_2$ has a complete $\alpha$-labeling with critical number $m$. $\square$

**Remark:** We note that a graceful tree $T$ with $m$ edges will have a vertex, say $v$, where $v$ has $m$ as a label. We now note that the above proof shows that a double of $T$ at $v$ has a complete $\alpha$-labeling with critical number $m$ where $v$ and its isomorphic copy, call it $v'$, have the labels $m$ and $2m + 1$, i.e. the critical number and the new maximum label. We will use this result later in Section 4 when constructing new classes of graceful trees.

The results that have been developed thus far enable us to state the following additional results on manners of joining graceful (or $\alpha$-labelable) graphs. These results represent just a selection of the results established by Ghosh [12] and will be stated without proof, but the general proof strategy for each of them involves showing that the resulting (bi)adjacency matrices of the graphs obtained after merging several other graphs together in some particular way remain completely graceful. Of the results presented here, only Proposition 3.8 will see use in Section 4.

**Proposition 3.8** (Ghosh, [12]). *Let $\{G_1, G_2, ..., G_r\}$ be a collection of graphs such that for each $i = 1, 2, ..., r$, $G_i = (V(G_i), E(G_i))$ has a complete $\alpha$-labeling with a critical number $k_i$, $|E(G_i)| = m_i$, and $V(G_i) \cap V(G_j) = \emptyset$ for all $i \neq j$. Let us denote a vertex in the graph $G_i$ with the label $\lambda$ by $\lambda_i$ and for convenience we denote $(k_i)_i$ by $k_i$ and $(m_i)_i$ by $m_i$. Let $G = (V(G), E(G))$ be a graph obtained by joining the vertices $k_i$ and $m_{i+1}$ with an edge for each $i = 1, 2, ..., r - 1$. Then $G$ has a complete $\alpha$-labeling with a critical number $k = k_1 + k_2 + \cdots + k_r + r - 1$.*

In essence, given a collection of graphs that all have a complete $\alpha$-labeling (and so are thus

graceful), we may chain these graphs together by connecting consecutive graphs by edges that adjoin the vertex labeled with the critical number of one graph to the vertex with the maximum label in the other. The resulting graph then has a complete $\alpha$-labeling, and so remains graceful.

**Proposition 3.9** (Ghosh, [12]). *Let $\{G_1, G_2, ..., G_r\}$ be a collection of graphs defined as in Proposition 3.8. Let $G = (V(G), E(G))$ be a graph obtained by joining the vertices $m_i$ and $m_{i+1}$ with an edge for each odd $i \in \{1, 2, ..., r-1\}$ and joining the vertices $k_i$ and $k_{i+1}$ with an edge for each even $i \in \{2, 3, ..., r-1\}$. Then $G$ has a complete $\alpha$-labeling with a critical number $k = k_1 + k_2 + \cdots + k_r + r - 1$. Moreover, if $m_i - k_i = m_{i+1} - k_{i+1}$ for all even $i \in \{2, 3, ..., r-1\}$, then the graph $H$ has a complete $\alpha$-labeling with a critical number $k = k_1 + k_2 + \cdots + k_r + r - 1$, where the graph $H$ is obtained by joining the vertices $m_i$ and $m_{i+1}$ with an edge for each $i = 1, 2, ..., r-1$.*

**Proposition 3.10** (Ghosh, [12]). *Let $\{G_1, G_2, ..., G_r\}$ be a collection of completely graceful graphs where $G_i = (V(G_i), E(G_i))$ with $|E(G_i)| = m_i$ for all $i = 1, 2, ..., r$ and $V(G_i) \cap V(G_j) = \emptyset$ for all $i \neq j$. Suppose that $m_1 = m_2 = \cdots = m_r$. For each $i = 1, 2, ..., r-1$, let $G_i'$ be an isomorphic copy of $G_i$. Let us denote a vertex in the graph $G_i$ with the label $\lambda$ by $\lambda_i$ and the corresponding vertex of $G_i'$ by $\lambda_i'$. Let $G = (V(G), E(G))$ be a graph obtained by joining a new vertex, say $v$, with each $m_i$ for $i = 1, 2, ..., r$ and with each $m_i'$ for $i = 1, 2, ..., r-1$. Then $G$ is a graceful graph with a graceful labeling in which the vertex $v$ has the maximum label.*

## 3.2 Transfers

As mentioned previously, Hrnčiar and Haviar are responsible for the development of a transfer technique for building graceful trees which was utilized heavily in their proof that all diameter five trees are graceful. This technique is the other primary influence of my efforts to establish new classes of graceful trees and will be discussed here.

**Definition 3.11** (Hrnčiar and Haviar, [14]). A $u \to v$ *transfer* is a transfer of end-edges

from vertex $u$ to vertex $v$. For brevity, a $u \to v$ transfer followed by a $v \to w$ transfer may be denoted by $u \to v \to w$, etc. A transfer is called a *transfer of the first type* if the end-vertices of the transferred end-edges are $k, k+1, ..., k+m$ for some $k, m$ where $u+v = k+(k+m)$ (and $k+(k+m) = k+1+(k+m-1) = k+2+(k+m-2) = ...$). A transfer is a *transfer of the second type* if the end-vertices of the transferred end-edges can be grouped into two groups $k, k+1, ..., k+m$ and $l, l+1, ..., l+m$, where $u+v = k+l+m$ (and $k+(l+m) = k+1+(l+m-1) = k+2+(l+m-2) = ...$).

The following examples illustrate how these transfer techniques work in practice.

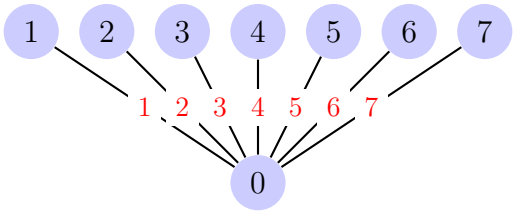


Figure 3.4: A graceful star T.

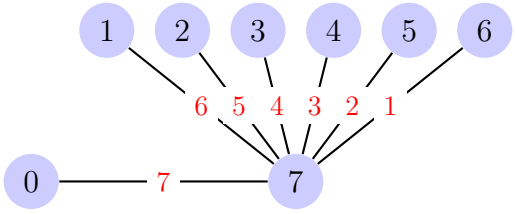


Figure 3.5: A graceful tree $T_1'$, the result of performing a $0 \to 7$ transfer of the first type on T, transferring the end-vertices 1,2,3,4,5, and 6 (where $k = 1, m = 5$ in the above definition).

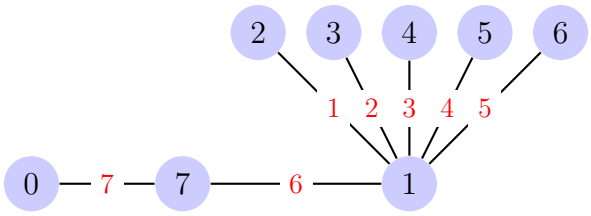


Figure 3.6: A graceful tree $T_1''$, the result of performing a $7 \to 1$ transfer of the first type on $T_1'$, transferring the end-vertices 2,3,4,5, and 6 (where $k = 2, m = 4$ in the above definition).

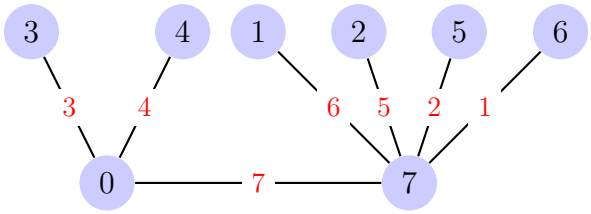


Figure 3.7: A graceful tree $T_2'$, the result of performing a $0 \to 7$ transfer of the second type on T, transferring the end-vertices 1,2,5, and 6 (where $k = 1, l = 5, m = 1$ in the above definition).

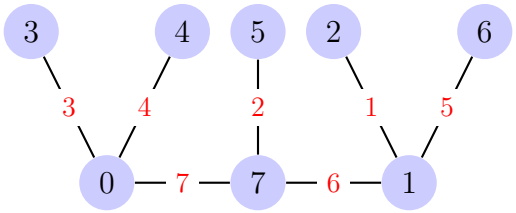


Figure 3.8: A graceful tree $T_2''$, the result of performing a $7 \to 1$ transfer of the second type to $T_2'$, transferring the end-vertices 2 and 6.

We now define a more sophisticated sequence of transfers below. This technique will be utilized in Section 4 when proving the gracefulness of a new class of trees.

**Definition 3.12** (Hrnčiar and Haviar, [14])**.** We say that the sequence of transfers $0 \to n \to 1 \to n-1 \to 0 \to n \to 1 \to n-1 \to 2$ where each transfer is of the first type is a *backwards double 8 transfer of the first type.*

## 3.3 Computational Approaches

In this section we discuss some of the computational approaches that have been utilized by Fang [10] and others [13] [1] in efforts to demonstrate the gracefulness of all trees with sufficiently few vertices. The most basic approach for finding graceful labelings of an arbitrary tree is to execute a simple brute force search for suitable labelings. For a tree with $n$ vertices, there are $n!$ many ways to assign vertex labels using the labels $\{0, 1, ..., n-1\}$. For small trees, it is a reasonable task for a computer to search through all possible labelings to find which are graceful. As $n$ becomes even moderately large, such a brute force method becomes infeasible. For example, for $n$ as small as 15, this translates to 1,307,674,368,000 possible labelings that need to be enumerated through for each tree of 15 vertices. This demonstrates the necessity of more sophisticated methods.

This brute force approach can be slightly modified to yield a more efficient search strategy. Rather than labeling all the vertices before checking whether the labeling is graceful, a labeling can incrementally be checked for gracefulness as each vertex is labeled. More specifically, when labeling a new vertex $v$, we consider all new induced edge labels before proceeding to the next vertex. If any of the induced edge labels after this step are duplicates, then without even considering the remaining unlabeled vertices we know we cannot produce a graceful labeling, and so we instead attempt to assign a different label to $v$. This method is much more efficient, as it will discard a large proportion of possible labelings that had no chance of being graceful at an early stage. However, there is still room for improvement.

The above observation suggests that rather than focusing on assigning unique vertex labels one at a time and checking at each step whether any of the induced edge labels are repeats, perhaps it would be more efficient to instead focus on obtaining the unique induced edge labels and at each step consider which remaining vertex label(s) can yield the desired induced edge label. We also note that there are fewer ways to obtain the larger induced edge labels, suggesting that these labels should be the priority. These observations motivate the approach utilized by Horton [13] and refined by Fang [10], which we now describe.

Horton's algorithm begins with assigning 0 as the label to the root of a tree $T$, where

$T$ has $n$ vertices. We then seek to obtain the induced edge labels in decreasing order from $n-1$ to 1. Thus, we first assign a neighbor of the root with the label $n-1$. We then assign an unlabeled vertex that is adjacent to either of the two labeled vertices with a label that yields an induced edge label of $n-2$, if possible. We repeat this process until all induced edge labels have been obtained or until we hit a dead end, at which point we backtrack to a previously valid state and try a different labeling. In order to prevent this algorithm from running too long in certain extreme cases, a threshold may be set on the number of backtracks that are permitted before the algorithm terminates. Overall this search strategy is efficient, but fixing a label of 0 for the root leaves out many potential solutions and in some cases this can result in the algorithm failing to find a graceful labeling (or taking an incredibly long time to find one) when one exists.

Another approach is the hill-climbing search developed by Aldred and McKay [1] and again refined by Fang [10]. This approach turns the search for a graceful labeling of a given tree into a combinatorial optimization problem, where the goal is to find an extremum of a specific evaluation function in a discrete domain in an attempt to answer a particular decision problem, which in this case is whether or not a given tree is graceful. The evaluation function used by Aldred and McKay is defined as follows: given a tree $T$ and labeling $L$, define $z(T, L)$ to be the number of distinct edge labels. The goal is thus to maximize this function by finding a labeling $L$ for $T$ such that $z(T, L) = n - 1$. Their algorithm begins by selecting a random labeling of $T$. If it maximizes the evaluation function then a graceful labeling has been found and we are done. Otherwise for every pair of vertices we switch their labels if it will increase the value of the evaluation function. If the labeling is unchanged after this process, we pick a random pair of vertices and swap their labels, provided we have not recently made that swap at a previous iteration of this step. We then repeat this process of considering possible label swaps between pairs. Hill climbing algorithms like this can get caught in local extrema, and so if this is detected then the algorithm is reset by picking another random labeling to start with.

As mentioned previously, Fang refined both of the above approaches and combined them

to create a hybrid search algorithm. In refining the backtracking algorithm, Fang experimentally determined a threshold to be set on the number of backtracks to be permitted in terms of $n$, where $n$ is the number of vertices in the tree being examined. Other optimizations are related to data representation. For example, Fang maintains a table of labels that have already been assigned and a linked list of vertices that have not yet been assigned a label but are adjacent to a vertex that has been assigned with a label. Furthermore, while maintaining this linked list, Fang considers the symmetry of isomorphic vertices and in doing so makes sure to exclude redundant labelings.

The hill-climbing search strategy was optimized as well, first by using a different evaluation function. The evaluation function utilized by Fang for a tree $T = (V(T), E(T))$ and a labeling $f$ is given by

$$Eval(f) = \sum_{i \in \{1,\ldots,n-1\} \setminus Im(g)} i,$$

where

$$Im(g) = \{|f(x) - f(y)| : xy \in E(T)\}.$$

The goal is to minimize this evaluation function, which will obtain a minimum of 0 only when $f$ is a graceful labeling. An advantage of this evaluation function is that it recognizes differences between the possible induced edge labels of $1, 2, ..., n-1$. We note that for $k \in \{1, 2, ..., n-1\}$, there are $k$ ways to obtain an induced edge label of $n-k$, i.e. there are fewer ways to obtain the larger induced edge labels. It thus makes sense that a larger penalty should occur due to a lack of large induced edge labels compared to smaller ones, and this is true of the evaluation function proposed by Fang.

Another improvement made by Fang is determining empirically an optimal value (in terms of the number of vertices in the tree) for the number of random modifications that should be tried at each step. Similarly, in attempting to avoid the problem of getting trapped in a local minimum, Fang's algorithm keeps track of some of the recently attempted modifications and forbids those that do not result in a graceful labeling, where the number of modifications to forbid is again determined empirically. One further improvement and addition to Fang's probabilistic search is the use of ideas from simulated annealing in an effort to again combat

the local minimum problem. At each step the algorithm, with an empirically determined probability, will also accept modifications that temporarily worsen the solution, which allows the algorithm to escape local minima and pursue new solutions. Fang's hybrid algorithm is then simply a combination of these two improved approaches. The hybrid algorithm first attempts to find a graceful labeling using the backtracking approach. If it is unable to do so before reaching the maximum allowed number of backtracks, which empirically only occurs in a small number of cases, it then transitions to the hill-climbing search. This approach proved to be successful, performing better than either the backtracking method or the hill-climbing method did alone.

In order to use any of the above approaches to prove the gracefulness of all trees of a given size, it is first necessary to enumerate all trees of that size. An algorithm for doing so was established by Wright et al. [21], summarized by Dinneen [8], and will be discussed briefly here. We first make the following definition.

**Definition 3.13.** Let $(T, v)$ be a rooted tree with root $v$. We say that a *layout* of $(T, v)$ is the lexicographically greatest distance sequence $L(T, v) = [l_1, l_2, ..., l_n]$ found when considering all preorder traversals, i.e. depth first searches, of $T$ beginning from $v$.

For clarity, we include an example below. Note that the "labels" given to each vertex represent their distance from the root, not labels in the sense that we have considered in the rest of this paper. The root in the example below is the vertex labeled with 0.

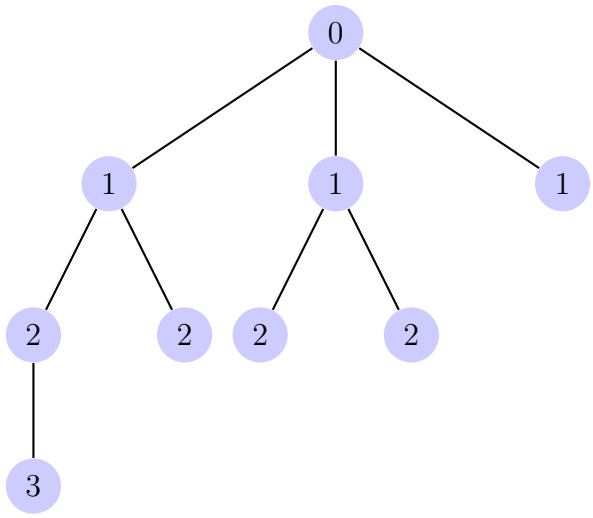


Figure 3.9: A rooted tree with layout [0,1,2,3,2,1,2,2,1].

We now outline how any tree can be regarded as a rooted tree (and thus be assigned a layout). To do so, we make some more definitions.

**Definition 3.14.** Let $T$ be a tree. We say that the *center* of $T$ is the set of vertices whose maximum distance from the other vertices is least.

**Remark:** It is always the case for trees that the center will contain either one or two vertices. When the center contains two vertices, these vertices will be adjacent. To find the center of a tree $T$, simply delete the leaves of $T$ to obtain a subtree $T'$, then delete the leaves of $T'$, etc. until we are left with either a single vertex or a single edge.

**Definition 3.15.** Let $T$ be a tree with center $C$. We define the *primary root* of $T$ in the following manner. If $C$ is a singleton $\{c_1\}$, then $c_1$ is the primary root of $T$. Otherwise, $C = \{c_1, c_2\}$. Let $(T_1, c_1)$ and $(T_2, c_2)$ be the two rooted trees obtained after removing the edge $c_1c_2$ from $T$. If $|V(T_1)| < |V(T_2)|$, or $|V(T_1)| = |V(T_2)|$ and $L(T_1, c_1) < L(T_2, c_2)$, then we say $c_2$ is the primary root of $T$. Otherwise, $c_1$ is the primary root of $T$.

We now may view any tree $T$ as a rooted tree $(T, r)$, where $r$ is the primary root of $T$. With this in mind, we define a successor function due to Beyer and Hedetniemi [5] that given a rooted tree of $n$ vertices will produce a different (non-isomorphic) tree of $n$ vertices. Let

$L = [l_1, l_2, ...l_n]$ be a layout of a rooted tree $T$ with $n$ vertices. Let $p$ be the largest integer subscript such that $l_p \neq 1$ and let $q$ be the largest integer subscript such that $q < p$ and $l_q = l_p - 1$. The successor function $s(L) = [s_1, s_2, ..., s_n]$ of $L$ is defined by:

$$s_i = \begin{cases} l_i & 1 \leq i < p \\ s_{i-(p-q)} & p \leq i \leq n \end{cases}$$

To use this successor function to enumerate through all trees of $n$ vertices, we begin with a path on $n$ vertices, i.e. $P_{n-1}$, with layout $[0, 1, 2, ..., n-1]$ and apply the successor function repeatedly until arriving at the last rooted tree, a star with $n$ vertices and with layout $[0, 1, 1, ..., 1]$. In order to ensure that this method works correctly and does not produce repeats, there is a necessary and sufficient test given by Wright et al. [21] to determine when a layout represents a tree rooted at its primary root, and in particular cases when a layout yielded by the successor function $s$ fails this test certain adjustments can be made to produce the correct successor. We will not discuss the specifics of this test nor the adjustments that may need to be made to a given layout here, but they can be found in [8] and [21].

# 4 Findings and New Techniques

## 4.1 Answers to Ghosh

### 4.1.1 Diameter Four Trees with Central Vertex of Even Degree

In the conclusion of Ghosh's paper [12], he posits a couple of questions regarding graceful tree labelings. One of these questions is whether all diameter four trees with a central vertex of even degree have a graceful labeling such that the label of the central vertex is maximum, similar to the result obtained for diameter four trees with a central vertex of odd degree in Lemma 2.21. As it turns out, the answer to this question is no, and it can be seen with a fairly simple counterexample which we shall work through below.

Consider the following rooted tree with a central vertex of degree 2.

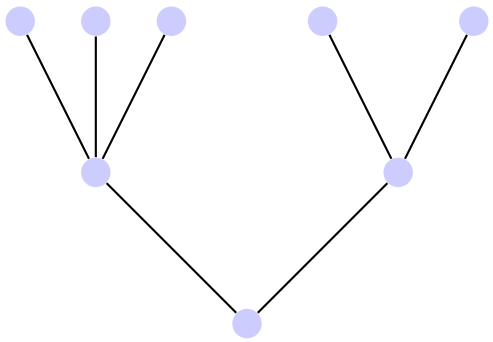

Figure 4.1: A rooted tree T with a central vertex of degree 2.

To label this in the desired manner, we would first assign the maximum label to the central vertex, which in this case would be 7 since T has eight vertices. For convenience, after labeling a vertex we will refer to that vertex by its label. In order for an induced edge label of 7 to be present, we must label one of 7's two neighbors with 0. Suppose we label its right neighbor with 0. We now must produce an induced edge label of 6 somewhere in T. This can be achieved by either labeling 7's left neighbor with 1, or assigning 6 as a label to either of 0's neighbors. We will first attempt assigning 1 to 7's left neighbor. Our current progress can be seen in the figure below.

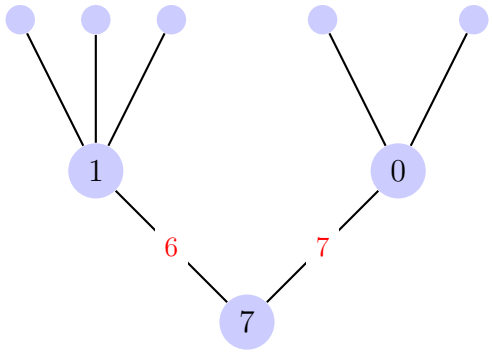


Figure 4.2: The first partial labeling of T.

We now note that neither of 0's neighbors can be assigned the label of 6, as this would repeat 6 as an induced edge label, so a neighbor of 1 must be assigned the label 6, yielding an induced edge label of 5. Similarly, now we cannot assign 5 to either of 0's neighbors, and so we must assign the label 5 to either of 1's two remaining neighbors, yielding an induced edge label of 4. Once more, 4 cannot be assigned to a neighbor of 0, so we assign it to 1's remaining neighbor, giving an induced edge label of 3. At this point, we are now stuck, as only 0's neighbors remain to be labeled, and yet assigning the only two available numbers will not yield a graceful labeling. We thus conclude that we cannot assign 7's left neighbor with the label 1 in this situation.

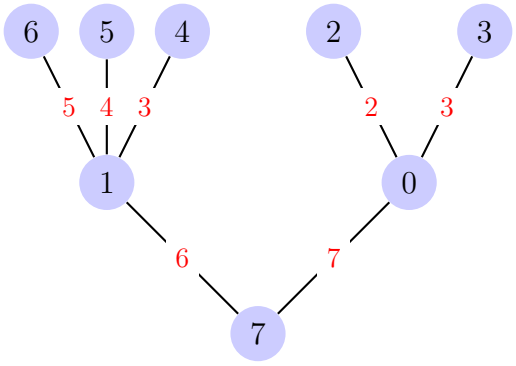


Figure 4.3: The point at which our first attempt to gracefully label T fails.

Reverting back to an earlier step, we now attempt to obtain an induced edge label of 6 by instead labeling one of 0's neighbors with a 6. The next induced edge label we seek to

obtain is 5, which can be achieved either by assigning a label of 2 to 7's left neighbor or a label of 5 to 0's remaining neighbor. We first consider assigning a label of 2 to 7's left neighbor. Our current progress is shown below.

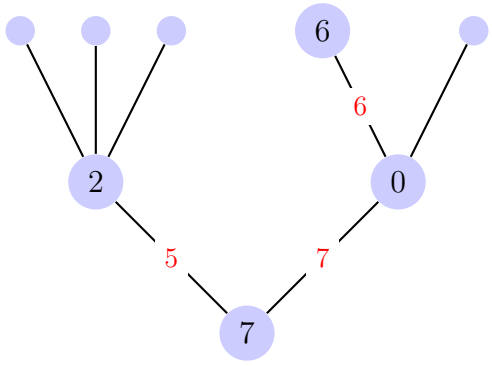


Figure 4.4: A second partial labeling of T.

We now wish to produce an induced edge label of 4. This can only be obtained by assigning a label of 4 to 0's remaining neighbor. But now we must assign the labels 1,3, and 5 to 2's neighbors, and so in particular the labels of 1 and 3 result in repeating the induced edge label of 1 and hence this labeling is not graceful. So, we conclude that we cannot assign 7's left neighbor the label of 2 in this situation.

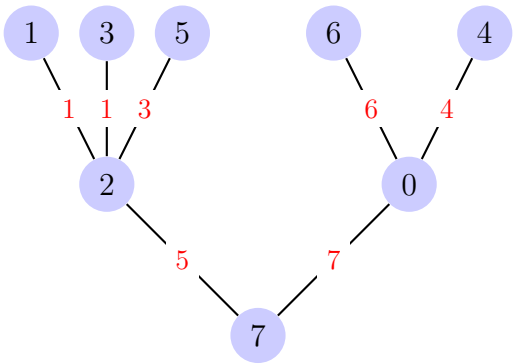


Figure 4.5: The point at which our second attempt to gracefully label T fails.

Instead, once again reverting to a previous step, we assign 0's remaining neighbor with a label of 5, as pictured below.

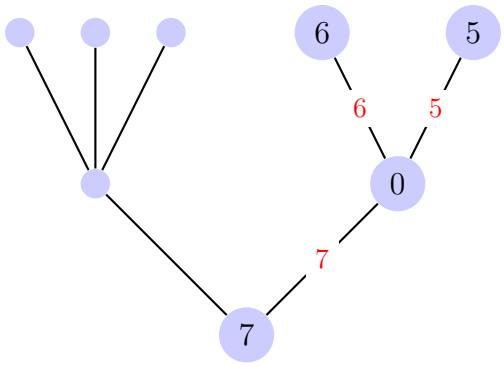


Figure 4.6: A third partial labeling of T.

We now wish to obtain an induced edge label of 4, and the only way to do so is to assign 7's left neighbor with the label 3. But again we are stuck, as the remaining labels for 3's neighbors are 1,2, and 4, and so in particular the 2 and 4 result in repeating the induced edge label of 1 and hence this labeling is not graceful.

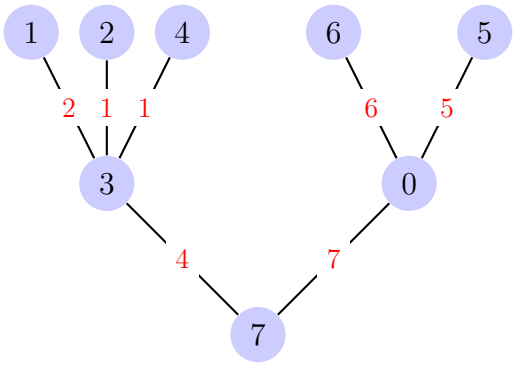


Figure 4.7: The point at which our third attempt to gracefully label T fails.

We have now shown that if 6 is assigned as a label to a neighbor of 0, then there is no way to obtain a graceful labeling, and so neither of 0's neighbors can be assigned a label of 6. Our entire argument thus far now demonstrates that 0 cannot be assigned as a label to 7's right neighbor. We would next attempt to assign 0 as a label to 7's left neighbor, and in a similar fashion as above it can be shown that no graceful labeling can result. We must therefore conclude that no graceful labeling of T exists where the central vertex is assigned the maximum label.

A logical next question to ask would be under what circumstances does a diameter four tree with a central vertex of even degree have a graceful labeling where the central vertex is given the maximum label, and this is exactly the question that I first explored. We begin with the following result.

**Proposition 4.1.** *Let $T$ be a diameter four tree with a central vertex of degree 2 and $4k+1$ vertices for some $k \in \mathbb{N}$. Then $T$ has a graceful labeling where the central vertex has the maximum label.*

*Proof:* Since $T$ has $4k+1$ vertices, the maximum vertex label to be used in a graceful labeling of $T$ is $4k$. We assign $4k$ to the central vertex, and 0 and $2k$ to its two neighbors. If $T$ is a symmetrical tree, we may label it as shown previously in Section 2.5, assigning the labels $2k+1, 2k+2, ..., 4k-1$ to the neighbors of 0 and the labels $1, 2, ..., 2k-1$ to the neighbors of $2k$. Otherwise, a graceful labeling of $T$ can be obtained by beginning with this graceful labeling of the symmetrical tree and performing the particular $2k \to 0$ transfer required. More specifically, if 0 and $2k$ each have an odd yet unequal number of leaves, a graceful labeling of T can be obtained by performing a $2k \to 0$ transfer of the second type. Otherwise, if 0 and $2k$ each have an even number of leaves, then a graceful labeling of T can be obtained by performing a $2k \to 0$ transfer of the first type. $\square$

This idea is illustrated below for all diameter four trees with 9 ($= 4k+1$ for $k = 2$) vertices and a central vertex of degree 2.

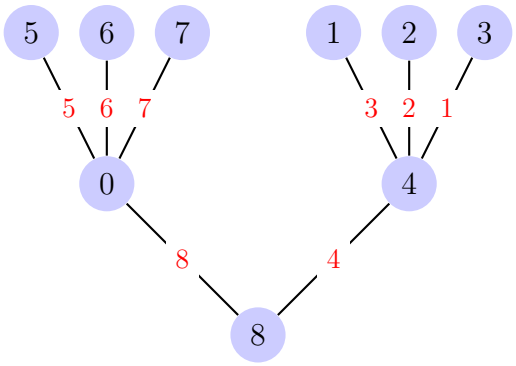


Figure 4.8: A graceful labeling of a diameter four tree with 9 vertices and a central vertex of degree 2 such that the central vertex has the maximum label.

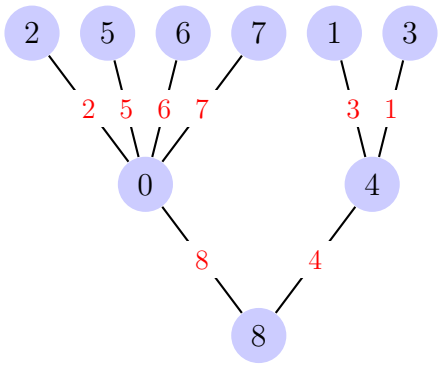


Figure 4.9: A graceful labeling of a diameter four tree with 9 vertices and a central vertex of degree 2 such that the central vertex has the maximum label.

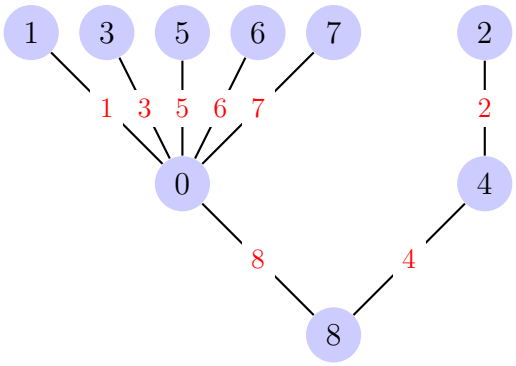


Figure 4.10: A graceful labeling of a diameter four tree with 9 vertices and a central vertex of degree 2 such that the central vertex has the maximum label.

In the above proof we began with a symmetrical tree with $4k+1$ vertices whose root had two branches and performed transfers to obtain the desired diameter four tree. A symmetrical tree with two branches at the root may instead have $4k+3$ vertices, which is the case where both neighbors of the root have $2k$ other neighbors i.e. an identical even number of neighbors that are not the root. We explore now under what circumstances a diameter four tree with $4k+3$ vertices and a central vertex of degree 2 is graceful where the central vertex is assigned the maximum label.

**Proposition 4.2.** *Let $T$ be a diameter four tree with a central vertex $v_0$ of degree 2 and $4k+3$ vertices for some $k \in \mathbb{N}$. Furthermore, suppose that each neighbor of $v_0$, call them $v_1$ and $v_2$, has an even number of leaves. Then $T$ has a graceful labeling where $v_0$ has the maximum label.*

*Proof:* Since $T$ has $4k+3$ vertices, the maximum vertex label to be used in a graceful labeling of $T$ is $4k+2$. We assign $4k+2$ to the central vertex $v_0$, and 0 and $2k+1$ to its two neighbors. As in the above proof of Proposition 4.1, if $T$ is a symmetrical tree, we may label it as shown previously in Section 2.5, assigning the labels $2k+2, 2k+3, ..., 4k+1$ to the neighbors of 0 and the labels $1, 2, ..., 2k$ to the neighbors of $2k+1$. Otherwise, a graceful labeling of $T$ can be obtained by beginning with this graceful labeling of the symmetrical tree and transferring the necessary number of pairs of leaves from $2k+1$ to 0 using a $2k+1 \to 0$ transfer of the

second type. □

We now illustrate this idea for all diameter four trees with 11 ($= 4k + 3$ for $k = 2$) vertices and a central vertex of degree 2.

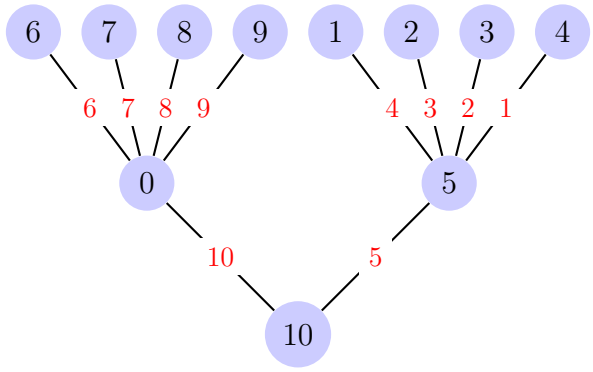


Figure 4.11: A graceful labeling of a diameter four tree with 11 vertices and a central vertex of degree 2 such that the central vertex has the maximum label.

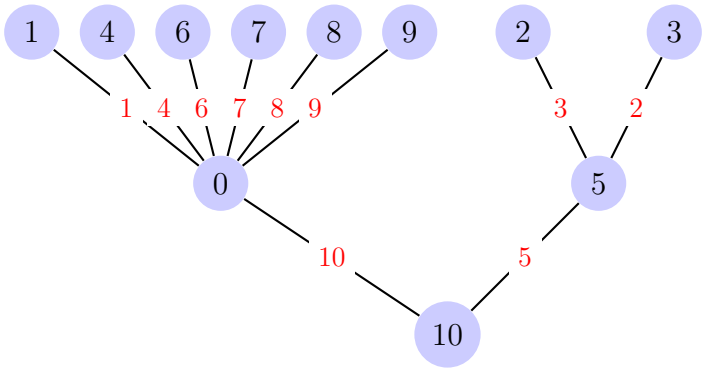


Figure 4.12: A graceful labeling of a diameter four tree with 11 vertices and a central vertex of degree 2 such that the central vertex has the maximum label.

The case remains where such a tree has an odd number of leaves at each neighbor. We cannot establish a graceful labeling of such trees where the central vertex has the maximum label using this approach, i.e. beginning with the canonical labeling of the initial symmetrical tree and then performing transfers. This is because we cannot transfer just a single vertex

from the $2k+1$ vertex as we could from the $2k$ vertex for trees with $4k+1$ vertices. The reason behind this is that $k$ is a neighbor of $2k$ in the initial symmetrical tree with $4k+1$ vertices, and so it can be transferred freely from $2k$ to 0 while preserving the induced edge label of $k$. There is no such option in the $4k+3$ case.

We therefore consider alternative labeling approaches. In particular, we consider the ways that we can label the two neighbors of the central vertex in a diameter four tree with $4k+3$ vertices and a central vertex of degree 2, given that we want the central vertex to have the maximum label, i.e. $4k+2$. We note that one of the neighbors, call it $v_1$, must have label 0, and so we only (possibly) have flexibility in how we choose to label the other neighbor, call it $v_2$. We now wish to determine which choices of label for $v_2$ are valid for at least one tree with $4k+3$ vertices.

We claim that $2k+l$ cannot be a label of $v_2$ in any such tree for $2 \leq l \leq 2k+1$. To see this, note that if $v_2$ has a label of $2k+l$, then the induced edge label of the edge connecting the central vertex and $v_2$ is $4k+2-(2k+l) = 2k+2-l$. Therefore it cannot be that the label of $2k+2-l$ is assigned to a neighbor of $v_1$ given that $v_1$ has label 0, and so instead $v_2$ must have a neighbor with label $2k+2-l$. The induced edge label of the edge connecting $v_2$ to this neighbor is then $2k+l-(2k+2-l) = 2l-2$. Again, we now have that $2l-2$ cannot be assigned as a label to a neighbor of $v_1$, and so $v_2$ must have another neighbor with label $2l-2$. This now induces an edge label of $2k+l-(2l-2) = 2k+2-l$ for the edge connecting $v_2$ and this neighbor, but this is the same edge label induced upon the edge connecting $v_2$ and the central vertex. Hence, there is no diameter four tree with $4k+3$ vertices where the central vertex has the maximum label and $v_2$ has the label $2k+l$ for $2 \leq l \leq 2k+1$ as claimed.

We note that $2k+l$ for $2 \leq l \leq 2k+1$ is never a divisor of $4k+2$, i.e. $4k+2 \not\equiv 0 \pmod{2k+l}$. This observation motivates the following result.

**Proposition 4.3.** *Let $n \geq 5,\ l \geq 2$. Then there exists a diameter four tree $T$ of $n$ vertices with a central vertex $v_0$ of degree 2 and with a graceful labeling $f$ where $v_0$ and its neighbors $v_1$ and $v_2$ satisfy $f(v_0) = n-1,\ f(v_1) = 0$, and $f(v_2) = l$ if and only if $n-1 \equiv 0 \pmod{l}$.*

*Proof:* We prove the forward direction by contrapositive. Let $n \geq 5,\ l \geq 2$ be given such that $n-1 \equiv r \pmod{l}$ for $r \geq 1$ where $r$ is the smallest possible value such that the congruence holds. We now wish to show that for any diameter four tree $T$ with a central vertex of degree 2 there does not exist a graceful labeling of $T$ such that the labels of the central vertex and its two neighbors are $n-1, 0$, and $l$ respectively. Suppose we attempted to construct such a tree $T$ with central vertex $v_0$ and neighbors $v_1$ and $v_2$ and a graceful labeling $f$ such that $f(v_0) = n-1,\ f(v_1) = 0$, and $f(v_2) = l$. We note that $r = n-1-ql$ for some $q$. Now, we have that $v_2$ must have leaves with the labels $n-1-l,\ n-1-2l, ...,\ n-1-ql = r$. To see this, we note that $|f(v_0) - f(v_2)| = n-1-l$, i.e. the induced edge label between $v_0$ and $v_2$ is $n-1-l$ and so $v_1$, whose label is 0, cannot have a leaf with label $n-1-l$ as otherwise this would repeat an induced edge label. Similarly, now this creates an induced edge label of $(n-1-l)-l = n-1-2l$ between $v_2$ and the leaf with label $n-1-l$, and so $v_1$ cannot have a leaf with label $n-1-2l$, etc.

We now note that since $v_2$ has a leaf with label $r$, the induced edge label between these two vertices is $l-r$, and so by similar logic $v_2$ must have a leaf with label $l-r$ which then induces an edge label of $l-(l-r) = r$. However, $v_2$ is already adjacent to a leaf with label $n-1-(q-1)l$, where the induced edge label between those two vertices is $n-1-(q-1)l-l = n-1-ql+l-l = n-1-ql = r$, and so $r$ has been repeated as an induced edge label. There is thus no way to construct a diameter four tree $T$ with a central vertex of degree 2 and a graceful labeling of $T$ where the central vertex and its neighbors have the labels $n-1, 0$, and $l$ respectively.

Now, suppose instead that $n-1 \equiv 0 \pmod{l}$. We again attempt to construct a diameter four tree $T$ with central vertex $v_0$ and neighbors $v_1$ and $v_2$ and a graceful labeling $f$ such that $f(v_0) = n-1,\ f(v_1) = 0$, and $f(v_2) = l$. We note that $n-1-ql = 0$ for some $q$, and so $n-1-kl = 2l$ for $k = q-2$. By the same logic as above, we have that $v_2$ must have $k$ leaves with the $k$ labels $n-1-l,\ n-1-2l, ...,\ n-1-kl = 2l$. We note now that the induced edge label between $v_2$ and the leaf with label $n-1-kl$ is $(n-1-kl)-l = 2l-l = l$, and so the only restriction this creates is that $v_1$ cannot have a leaf with label $l$. Since $l$ has

already been assigned to $v_2$, this does not present an issue. We thus can attach $n-k-3$ leaves to $v_1$ and assign them the remaining $n-k-3$ available labels between 0 and $n-1$, and the result is a diameter four tree of $n$ vertices with a central vertex $v_0$ of degree 2 and a graceful labeling $f$ where $v_0$ and its neighbors $v_1$ and $v_2$ satisfy $f(v_0) = n-1$, $f(v_1) = 0$, and $f(v_2) = l$. □

**Note:** The above proposition was for general diameter four trees with central vertex of degree 2, and so in particular it applies to such trees with $4k+3$ vertices as well.

Below are examples of the logic in the above proof for $n = 11$ and $l = 2, 3$.

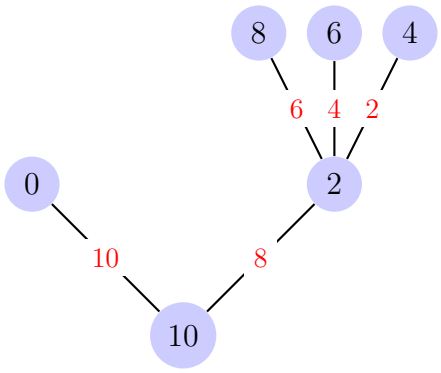


Figure 4.13: The labels that must be assigned to leaves of $v_2$ when $n = 11$, $l = 2$. Note that we may now attach leaves with labels 1,3,5,7, and 9 to $v_1$.

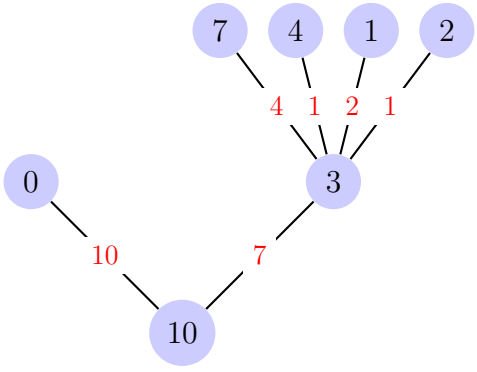


Figure 4.14: The labels that must be assigned to leaves of $v_2$ when $n = 11$, $l = 3$. Note that we are forced to repeat an induced edge label.

The above proposition characterizes which labels are valid for $v_2$ in diameter four trees with a central vertex of degree 2 as well as how many leaves must be attached to $v_2$ in each case. However, it does not characterize those trees that can be formed by first assigning all remaining leaves and labels to $v_1$ and then performing transfers when possible. Efforts in this direction could provide a complete summary of all diameter four trees with central vertex of degree 2 that can be labeled gracefully where the central vertex has the maximum label. We have also not given any consideration to diameter four trees with a central vertex of even degree $d$ for $d \geq 4$. We will discuss this briefly in the section Further Research Ideas.

### 4.1.2 Lobsters with Certain Pairwise Lobes

Ghosh also puts forth a class of lobsters that he believes may be graceful. More specifically, consider the following general form of a lobster of diameter at most five.

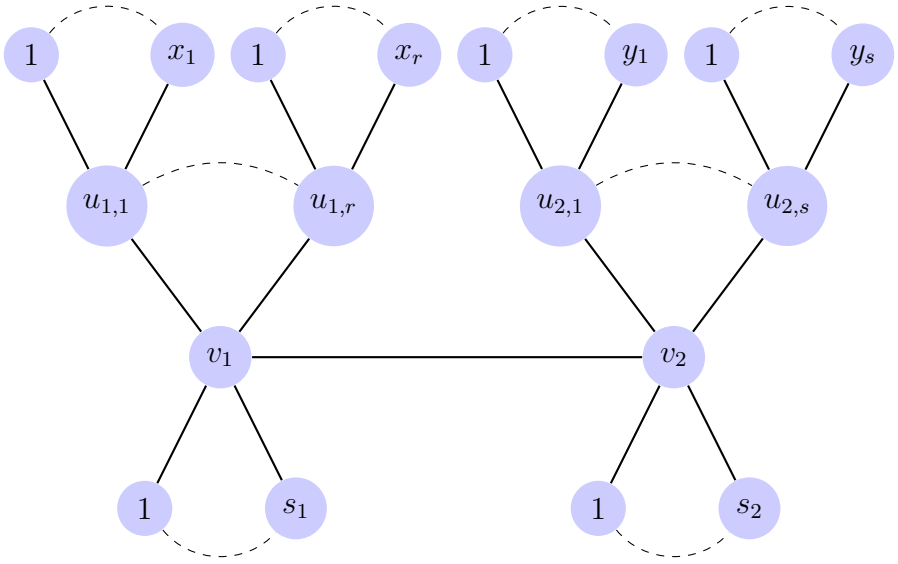


Figure 4.15: The general form of a lobster of diameter at most five.

The above lobster has central vertices $v_1$ and $v_2$, where $v_1$ has $r$ branches and each branch has $x_i$ leaves for each $i = 1, 2, ..., r$ and similarly $v_2$ has $s$ branches where each branch has $y_j$ leaves for each $j = 1, 2, ..., s$. Furthermore, $v_1$ and $v_2$ are adjacent to $s_1$ and $s_2$ leaves,

respectively. We say that the subtrees rooted at $v_1$ and $v_2$ are *lobes* of the lobster. Ghosh suggests that lobsters whose pairwise lobes satisfy the property $(*)$ that $s_1 = s_2 = 0$, $r = s$, $\sum_{i=1}^{r} x_i = \sum_{i=1}^{r} y_i = q$, and $r$ divides $q$ may be graceful. The motivation behind this suggestion comes from the fact that Ghosh proved that the lobster in Figure 4.15 where $s_1 = s_2 = 0$, $r = s$, and $x_i = y_i = l$ for each $i = 1, 2, ..., r$ and for some $l$ has an $\alpha$-labeling where $v_1$ is assigned the maximum label and $v_2$ is assigned the label of $k$, where $k$ is the critical number, and hence such lobsters are graceful. Ghosh calls such a lobster "trivially balanced." It follows by Proposition 3.8 that we may chain arbitrarily many trivially balanced lobsters together and the resulting lobster will remain both graceful and $\alpha$-labelable.

To obtain a lobster whose lobes satisfy the property $(*)$, Ghosh suggested that they may be constructed by starting with the corresponding trivially balanced lobster where $x_i = y_i = \frac{q}{r}$ for each $i = 1, 2, ..., r$ and then "transferring" leaves (not in the way defined by Hrnčiar and Haviar [14]) from one branch to another, where leaves are never transferred across lobes. The reason behind not transferring leaves across lobes is that the resulting labeling would no longer be an $\alpha$-labeling. When making a valid sequence of transfers in this context, each of the transfers that can be performed will not preserve gracefulness on their own, but they will preserve gracefulness when taken as a whole. For example, we may transfer a leaf from one branch to another and in the process exchange an induced edge label of $l_1$ for an induced edge label of $l_2$, followed by a separate transfer that exchanges an induced edge label of $l_2$ for an induced edge label of $l_1$. While these transfers are not entirely in the spirit of those defined by Hrnčiar and Haviar, we will nevertheless use their transfer notation.

Before considering this transfer approach, we combine our above work on diameter four trees with a central vertex of degree 2 and Propositions 3.2 and 3.8 to state the following result.

**Proposition 4.4.** *Let $T_1, T_2, ..., T_n$ be trees such that for each $i = 1, 2, ..., n$, $T_i$ is a tree of either the form given in Proposition* 4.1 *or Proposition* 4.2*, and let $T_1', T_2', ..., T_n'$ be isomorphic copies of $T_1, T_2, ..., T_n$. Then the lobster $L$ whose lobes in order are $T_1, T_1', T_2, T_2', ..., T_n, T_n'$,*

*i.e. whose pairwise lobes are isomorphic copies of $T_1, T_2, ..., T_n$, has an $\alpha$-labeling and hence is graceful.*

**Note:** The lobster constructed in the above proposition satisfies property $(*)$.

We now return to the transfer idea put forth by Ghosh. We present no formal proofs here, but rather examine the types of transfers that are possible for some small cases and consider the implications in regards to Ghosh's conjecture. We note here that a trivially balanced lobster is just a double of a symmetrical tree. We consider now the lobes of some trivially balanced lobsters along with their standard labelings (i.e. the labeling obtained from a double of a canonically labeled symmetrical tree). Since transfers must occur within a given lobe, in each case we will depict the lobes separately. As done in Figure 4.15, we will denote the root of branch $j$ in lobe $i$ by $u_{i,j}$, where the branches are ordered from left to right as depicted in the figures below. We begin with a trivially balanced lobster where each lobe has three branches and each branch has three leaves, i.e. $r = s = 3$ and $x_i = y_i = 3$ for each $i = 1, 2, 3$.

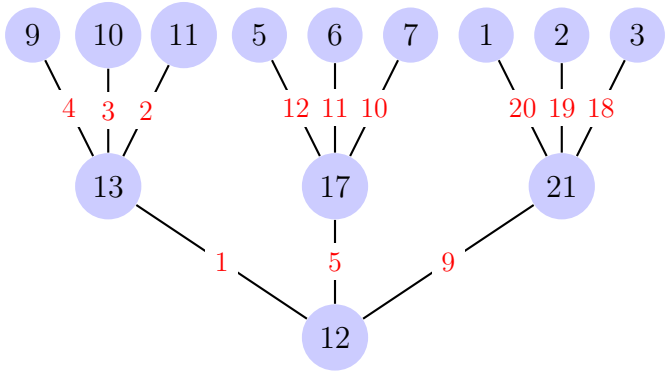


Figure 4.16: The first lobe of a trivially balanced lobster where each lobe has three branches and each branch has three leaves.

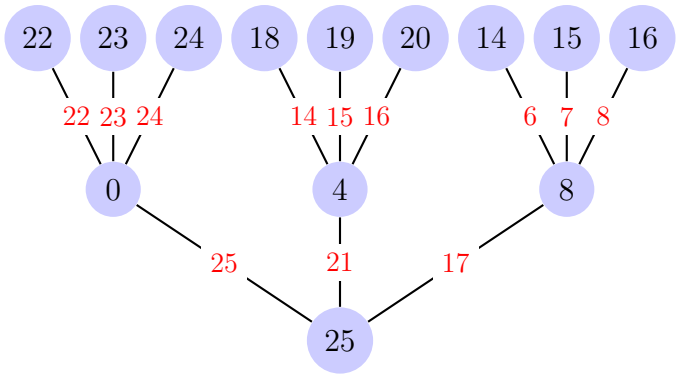


Figure 4.17: The second lobe of a trivially balanced lobster where each lobe has three branches and each branch has three leaves.

For initially balanced lobes, we observe that the pairs of transfers which are possible are $(u_{1,3} \to u_{1,2}, u_{2,2} \to u_{2,1})$, $(u_{1,2} \to u_{1,1}, u_{2,3} \to u_{2,2})$, and $(u_{1,2} \to u_{1,3}, u_{2,2} \to u_{2,3})$. We also note that there is symmetry between the first two pairs and that the third pair involves making an identical transfer in each lobe. We now consider a trivially balanced lobster where each lobe has four branches and each branch has three leaves, i.e. $r = s = 4$ and $x_i = y_i = 3$ for each $i = 1, 2, 3, 4$.

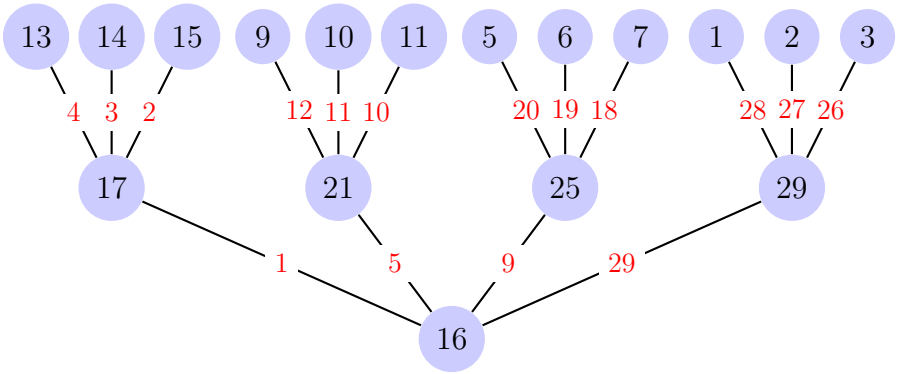


Figure 4.18: The first lobe of a trivially balanced lobster where each lobe has four branches and each branch has three leaves.

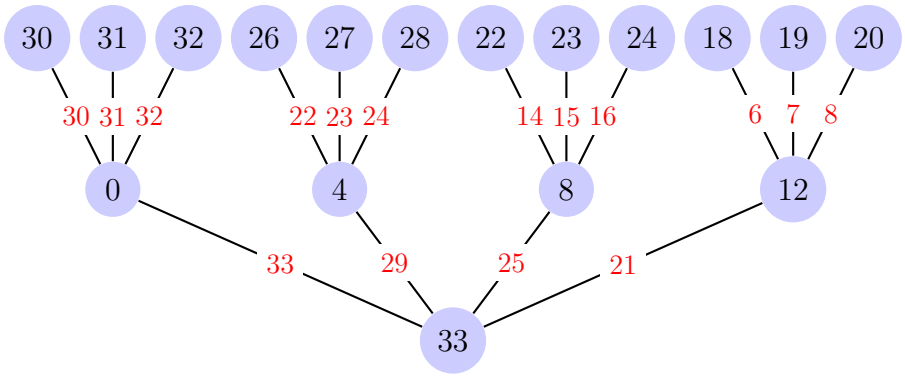


Figure 4.19: The second lobe of a trivially balanced lobster where each lobe has four branches and each branch has three leaves.

We observe now that the pairs of transfers that are initially possible are $(u_{1,4} \to u_{1,3}, u_{2,2} \to u_{2,1})$, $(u_{1,2} \to u_{1,1}, u_{2,4} \to u_{2,3})$, $(u_{1,3} \to u_{1,4}, u_{2,2} \to u_{2,3})$, $(u_{1,2} \to u_{1,3}, u_{2,3} \to u_{2,4})$, $(u_{1,3} \to u_{1,1}, u_{1,2} \to u_{1,4})$, $(u_{2,3} \to u_{2,1}, u_{2,2} \to u_{2,4})$, and $(u_{1,3} \to u_{1,2}, u_{2,3} \to u_{2,2})$. We again note the symmetries that are present; there is symmetry between the first and second pairs, the third and fourth pairs, and the fifth and sixth pairs. In the case of the fifth and sixth pairs, these transfer pairs are contained entirely in a single lobe. Furthermore, the seventh pair involves making an identical transfer in each lobe. We note as well the similarity in form of some of these transfer pairs with those from the 3 branch case above. If we denote the number of branches by $j$, we have that in each instance the transfer pairs $(u_{1,j} \to u_{1,j-1}, u_{2,2} \to u_{2,1})$ and $(u_{1,2} \to u_{1,1}, u_{2,j} \to u_{2,j-1})$ are possible. We also have that the transfer pairs $(u_{1,j-1} \to u_{1,j}, u_{2,2} \to u_{2,3})$ and $(u_{1,2} \to u_{1,3}, u_{2,j-1} \to u_{2,j})$ are valid in both cases.

It is likely that similar patterns would emerge when considering trivially balanced lobsters with a larger number of branches, and both the number of valid transfer pairs and the forms they take may in fact be recursively definable in terms of the number of branches. From looking at these possible transfers for these small cases, it is not immediately clear exactly what kind of graceful lobsters we can construct (in terms of the distribution of the leaves within each lobe) using this method of transferring leaves. Preliminary work suggests that this approach alone may not be sufficient to prove Ghosh's conjecture, but such a claim is

not substantiated here.

## 4.2 Graceful Trees Built By Transfers

In this section, I present some novel classes of graceful trees constructed using transfers. In the spirit of "caterpillars", "lobsters", "spiders", etc., these trees are given animal titles.

### 4.2.1 Crabs

**Definition 4.5.** Let $G$ be a tree such that its central vertex $v_0$ is of degree 3, and two of its neighbors, call them $v_1$ and $v_2$, each have $k$ neighbors that are not $v_0$ and $k-1$ of which are leaves, while the third neighbor of $v_0$, call it $v_3$, has degree 2, i.e. it has a single neighbor that is not $v_0$, and this neighbor is a leaf. Let the non-leaf neighbors of $v_1$ and $v_2$ that are not $v_0$ be denoted $u_1$ and $u_2$ respectively. We further suppose that $u_1$ and $u_2$ are of degree 2, and that their second neighbors (i.e neighbors other than $v_1$ and $v_2$, respectively) are leaves. We say that such a tree $G$ is a *crab leg with index k*. Furthermore, we consider a graph formed by joining two identical crab legs together by an edge connecting their central vertices to be a *crab*.

Consider the figure below to understand the inspiration behind this name.

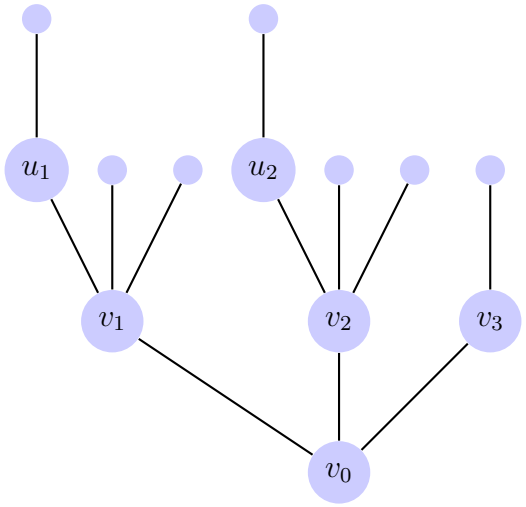


Figure 4.20: A crab leg graph with index 3.

**Remark:** We note that a crab leg graph with index $k$ has $2k + 7$ vertices.

**Proposition 4.6.** *Let $G$ be a crab leg graph with index $k$ and with central vertex $v_0$. Then there exists a graceful labeling $f$ of $G$ such that $f(v_0) = |V(G) - 1| = 2k + 6$, i.e. $v_0$ has the maximum label.*

*Proof:* Let $G$ be a crab leg graph. There are two cases to consider depending on whether $k$ is odd or even. We first consider the case where $k$ is odd. We note that $k = 2l + 1$ for some $l$, and so we note that $G$ has $2(2l + 1) + 7 = 4l + 9$ vertices. We begin by assigning $v_0$ the label $4l + 8$, $v_1$ the label 0, $v_2$ the label 1, and $v_3$ the label $4l + 7$. We attach leaves with the remaining labels $2, 3, ..., 4l + 6$ to 0 and will construct $G$ via transfers. First, make a $0 \to 4l + 7$ transfer of the first type, transferring the vertices $l + 2, l + 3, l + 4, ..., 3l + 5$ and leaving behind the $2l + 1 = k$ vertices labeled $2, 3, ...l + 1$ and $3l + 6, 3l + 7, ...4l + 6$. We then perform a $4l + 7 \to 1$ transfer of the first type, transferring the $2l + 3 = k + 2$ vertices $l + 3, l + 4, ..., 3l + 5$ and leaving the vertex $l + 2$. Finally, we make $1 \to 4l + 5$ and $1 \to 2l + 5$ transfers of the first type, transferring the vertices $2l + 3$ and $l + 3$ respectively.

We now consider the case when $k$ is even. We note that $k = 2l$ for some $l$, and so we note that $G$ has $2(2l) + 7 = 4l + 7$ vertices. We begin by assigning $v_0$ the label $4l + 6$, $v_1$ the label 1, $v_2$ the label $4l + 5$, and $v_3$ the label 0. We attach leaves with the remaining labels $2, 3, ..., 4l + 4$ to 0 and will construct $G$ via transfers as before. First, make a $0 \to 4l + 5$ transfer of the first type, transferring the vertices $2, 3, ..., 4l + 3$ and leaving behind the single vertex $4l + 4$. We then perform a $4l + 5 \to 1$ transfer of the second type, transferring the $2l + 2 = k + 2$ vertices $l + 2, l + 3, ..., 2l + 2, 2l + 4, 2l + 5, ..., 3l + 4$, and leaving the $2l = k$ vertices $2, 3, ..., l + 1, 2l + 3, 3l + 5, 3l + 6, ..., 4l + 3$. Finally, we make $1 \to 2l + 3$ and $1 \to 2l + 5$ transfers of the first type, transferring the vertices $l + 2$ and $l + 3$ respectively. □

We now demonstrate this labeling approach for $k = 3$ and $k = 4$.

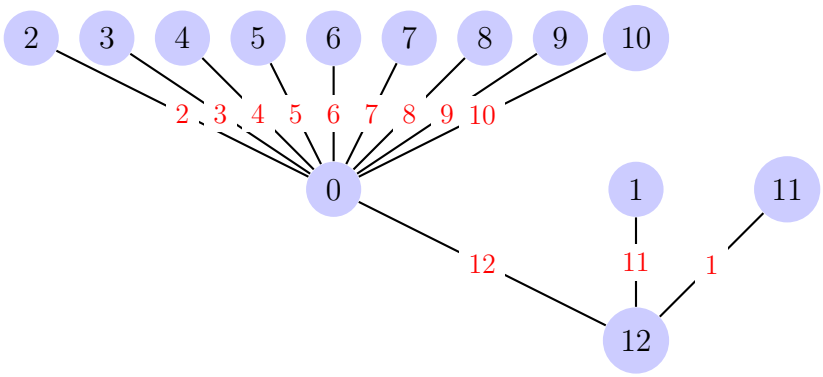


Figure 4.21: The starting point of our attempt to label the crab leg graph with index 3.

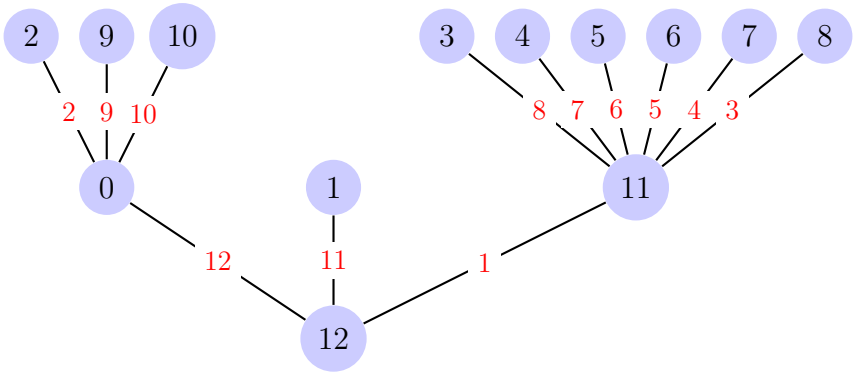


Figure 4.22: The result of performing a $0 \to 11$ transfer.

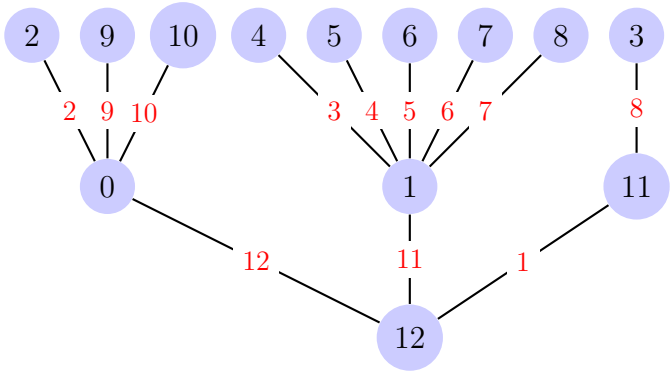


Figure 4.23: The result of performing an $11 \to 1$ transfer.

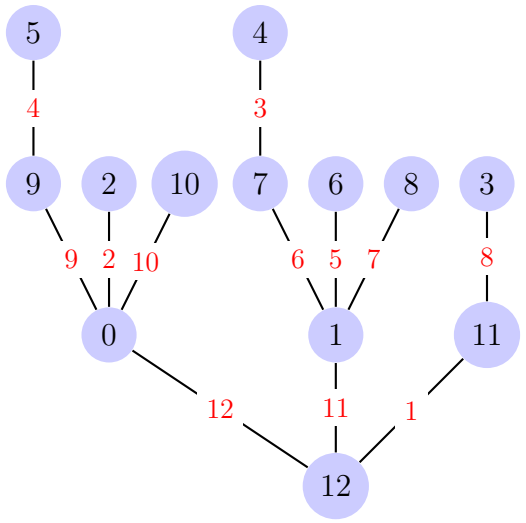


Figure 4.24: A graceful labeling for the crab leg graph with index 3.

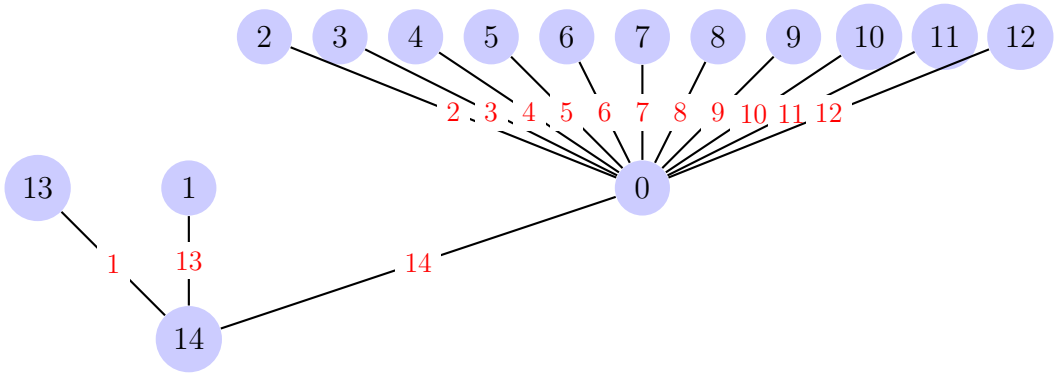


Figure 4.25: The starting point of our attempt to label the crab leg graph with index 4.

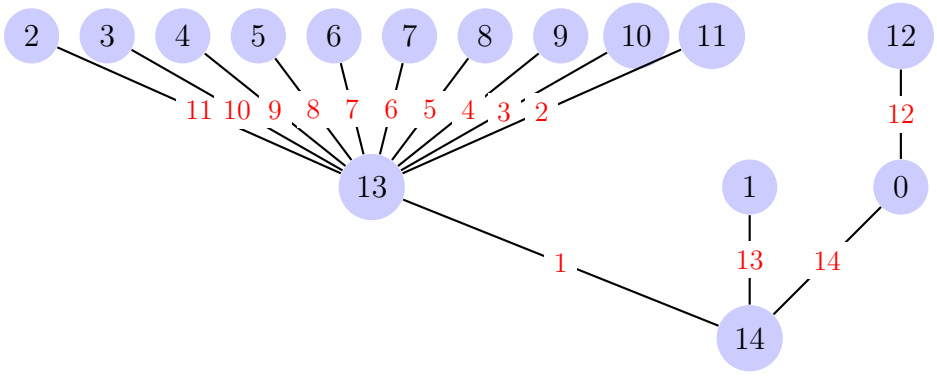


Figure 4.26: The result of performing a $0 \to 13$ transfer.

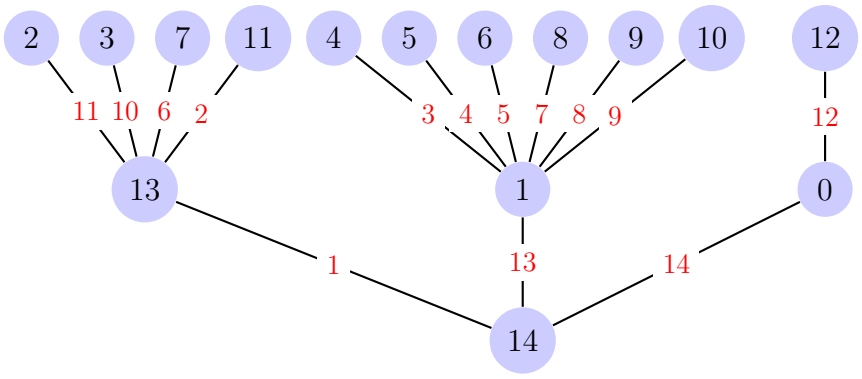


Figure 4.27: The result of performing a $13 \to 1$ transfer.

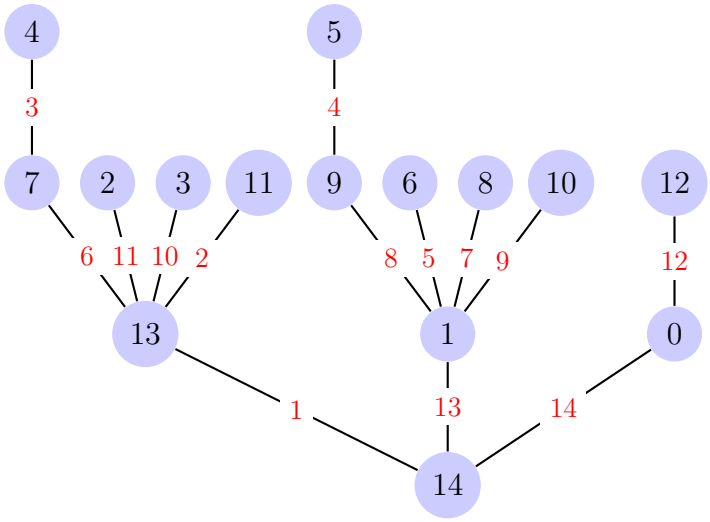


Figure 4.28: A graceful labeling for a crab leg graph with index 4.

The next result now follows immediately as a result of Proposition 3.2.

**Corollary 4.6.1.** *Let $G$ be a crab graph with $m$ edges. Then there exists an $\alpha$-labeling $f$ with critical number $k$ of $G$ such that the central vertices of the two crab legs are labeled with $k$ and $m$.*

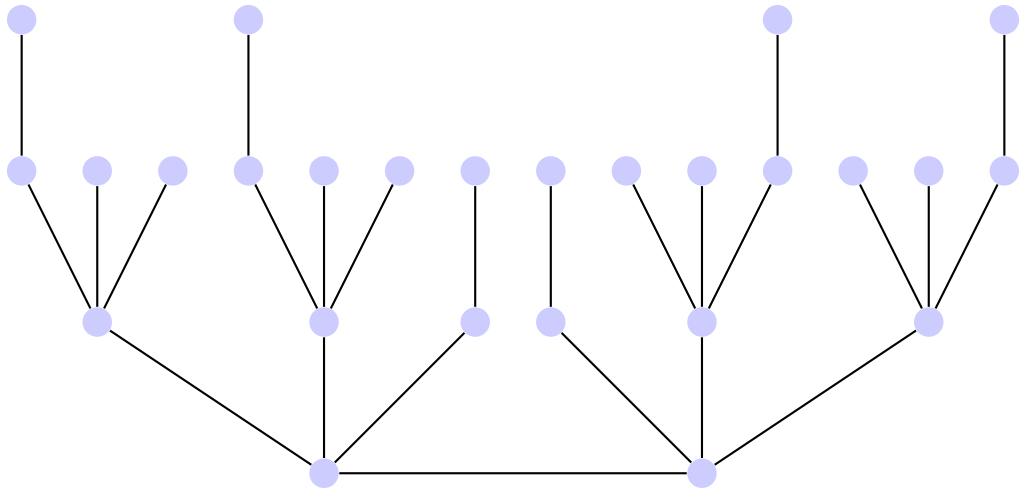

Figure 4.29: A crab graph where each leg has index 3 as shown in Figure 4.20.

It then follows by Proposition 3.8 that we may chain arbitrarily many crab graphs together and the resulting graph will be graceful, and in fact $\alpha$-labelable.

### 4.2.2 Butterflies

**Definition 4.7.** Let $G$ be a tree constructed by identifying three copies of $P_2$ and two copies of $P_3$ with the central vertex $v_0$ (that is, three copies of $P_2$ and two copies of $P_3$ are extending from $v_0$). Furthermore, attach a leaf to each of the three vertices that are not $v_0$ in one of the copies of $P_3$. Finally, let $v_0$ have an even number of leaves. We say that such a graph $G$ is a *butterfly wing*. Furthermore, we consider a graph formed by joining two identical butterfly wings together by an edge connecting their central vertices to be a *butterfly*.

As with crab leg graphs, consider the figure below to understand the motivation for this name.

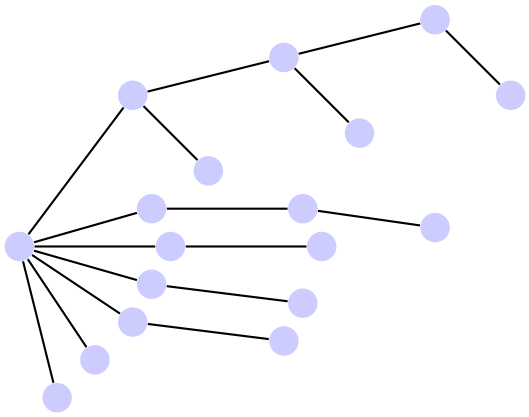

Figure 4.30: A butterfly wing graph where the central vertex has two leaves.

**Remark:** We note that a butterfly wing graph whose central vertex has $k$ leaves has $k+16$ vertices.

**Proposition 4.8.** *Let $G$ be a butterfly wing graph with central vertex $v_0$, where $v_0$ has $k$ leaves. Then there exists a graceful labeling $f$ of $G$ such that $f(v_0) = |V(G) - 1| = k + 15$, i.e. $v_0$ has the maximum label.*

*Proof:* We begin with a graceful star of $k+16$ vertices, where the central vertex has label 0 and the leaves have the labels $1, ..., k+15$. We begin a backwards double 8 transfer of the first type, i.e. a $0 \to k+15 \to 1 \to k+14 \to 0 \to k+15 \to 1 \to k+14 \to 2$ transfer of the first type. At each step of this transfer we make a maximal transfer, i.e. transferring every possible vertex at each step, with the exception of the second $k+15 \to 1$ transfer. At this step, we leave behind $3+k$ vertices. More specifically, we begin by making the following maximal transfers: we transfer $1, 2, ..., k+14$ from 0 to $k+15$, then transfer $2, 3, ..., k+14$ from $k+15$ to 1, then transfer $2, 3, ..., k+13$ from 1 to $k+14$, followed by transferring $2, 3, ..., k+12$ from $k+14$ to 0, and finally transferring $3, 4, ..., k+12$ from 0 to $k+15$. We then transfer the 7 vertices $5+\frac{k}{2}, 6+\frac{k}{2}, ..., 11+\frac{k}{2}$ from $k+15$ to 1, leaving behind the $3+k$ vertices $3, 4, ..., 4+\frac{k}{2}$ and $12+\frac{k}{2}, 13+\frac{k}{2}, ..., k+12$ as leaves of $k+15$. We finish this backwards double 8 transfer sequence by transferring $5+\frac{k}{2}, 6+\frac{k}{2}, ..., 10+\frac{k}{2}$ from 1 to $k+14$, and $6+\frac{k}{2}, 7+\frac{k}{2}, ..., 10+\frac{k}{2}$ from $k+14$ to 2.

After these transfers, we then perform a maximal $2 \to k+13 \to 3$ transfer, transferring $6+\frac{k}{2}, 7+\frac{k}{2}, 8+\frac{k}{2}$, and $9+\frac{k}{2}$ from 2 to $k+13$ and $7+\frac{k}{2}, 8+\frac{k}{2}$, and $9+\frac{k}{2}$ from $k+13$ to 3. We complete our construction of a graceful labeling for $G$ by transferring $7+\frac{k}{2}$ from 3 to $12+k$ and $8+\frac{k}{2}$ from 3 to 4. $\square$

**Remark:** Similar to those discussed in Section 4.1.2, the final two "transfers" in the above proof are not transfers in the traditional sense as defined by Hrnčiar and Haviar [14], as each individual transfer does not preserve the original induced edge label. The original edge between 3 and $7+\frac{k}{2}$ has an induced edge label of $4+\frac{k}{2}$ and after the transfer the new edge between $12+k$ and $7+\frac{k}{2}$ has an induced edge label of $5+\frac{k}{2}$, while the original edge between 3 and $8+\frac{k}{2}$ has an induced edge label of $5+\frac{k}{2}$ and after the transfer the new edge between 4 and $8+\frac{k}{2}$ has an induced edge label of $4+\frac{k}{2}$. When made together, these "transfers" preserve a graceful labeling.

We demonstrate this graceful labeling construction for the butterfly wing graph from Figure 4.30.

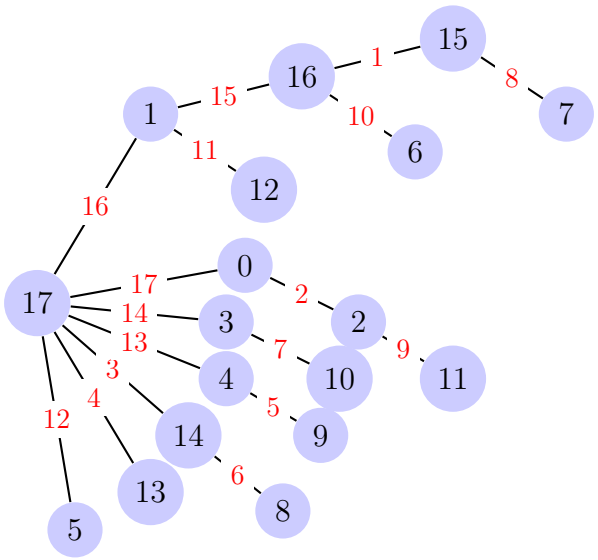


Figure 4.31: A graceful labeling for the butterfly wing graph from Figure 4.30.

The next result now follows immediately as a result of Proposition 3.2.

**Corollary 4.8.1.** *Let $G$ be a butterfly graph with $m$ edges. Then there exists an $\alpha$-labeling $f$ with critical number $k$ of $G$ such that the central vertices of the two butterfly wings are labeled with $k$ and $m$.*

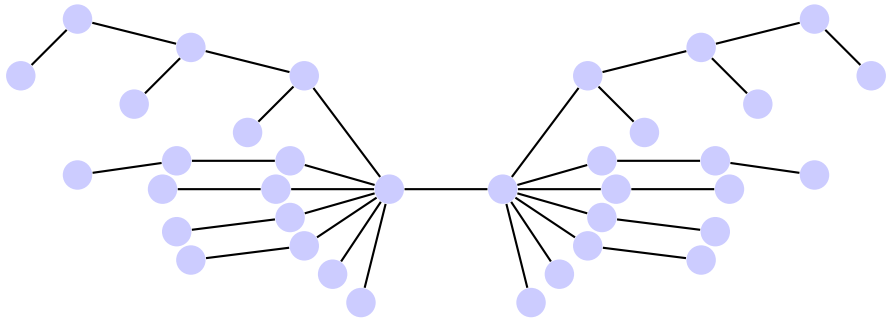

Figure 4.32: A butterfly graph where each wing is as shown in Figure 4.30.

It follows by Proposition 3.8 that we may chain arbitrarily many butterfly graphs together and the resulting graph will be $\alpha$-labelable, and hence graceful.

## 4.3 Graceful Expansions

The techniques in the following section are methods for "growing" larger graceful trees from an initial tree with a given graceful labeling. To the best of my knowledge, these techniques have not been discussed in the existing research on graceful trees.

**Definition 4.9.** Let $T$ be a graceful tree with a graceful labeling $f$. The *$k$-factor expansion of $T$ given $f$* is the tree that results from multiplying the labels of $T$ by $k$ and adding the remaining neighbors to the vertex labeled 0. More specifically, if $T$ has $n$ vertices, then multiplying the labels of $T$ by $k$ yields a new maximum label of $k(n-1) = kn - k$, and so this expanded tree must have $kn - k + 1$ vertices. We add the additional $kn - k + 1 - n = (k-1)n - k + 1$ vertices required as leaves of the vertex labeled 0 and assign these vertices with the labels that are not currently present in the tree, namely the labels corresponding to values in $\{0, 1, ..., kn - k\}$ that are neither 0 nor divisible by $k$. We denote the resulting tree by $T_f^k$.

Below is an example of a graceful tree $T$ with two labelings $f, g$ followed by their 2-factor expansions.

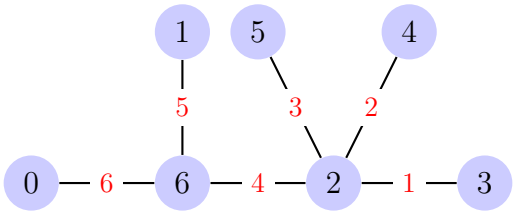


Figure 4.33: A graceful tree $T$ with a labeling $f$.

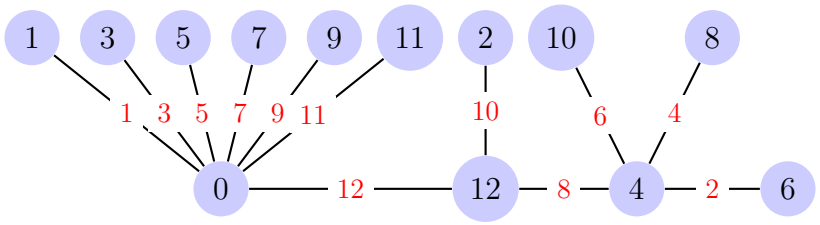


Figure 4.34: $T_f^2$, the 2-factor expansion of $T$ given $f$.

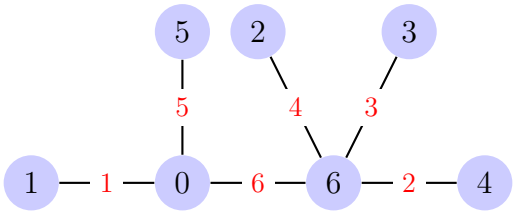


Figure 4.35: A graceful tree $T$ with a labeling $g$.

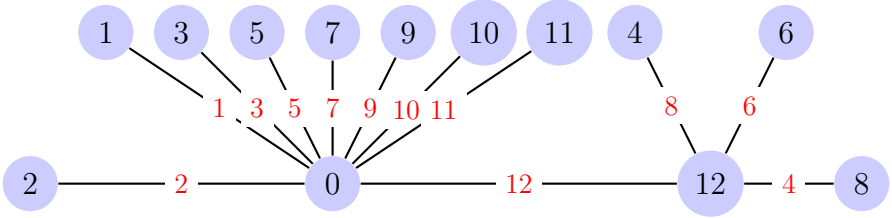


Figure 4.36: $T_g^2$, the 2-factor expansion of $T$ given $g$.

On their own, these expansions are not particularly powerful. For example, the tree $T$ above is a caterpillar graph, as are its 2-factor expansions given the labelings $f$ and $g$ respectively, and so we already know such trees can be labeled gracefully. The ability of these expansions to produce graceful labelings of non-trivial trees comes from the transfers that can be performed on these expansions. Consider the following trees and their accompanying graceful labelings obtained from $T_f^2$ and $T_g^2$ respectively by making several transfers.

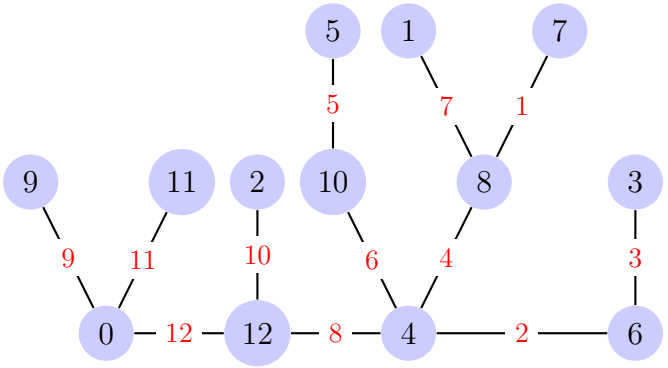


Figure 4.37: A gracefully labeled tree obtained from $T_f^2$ by making several transfers.

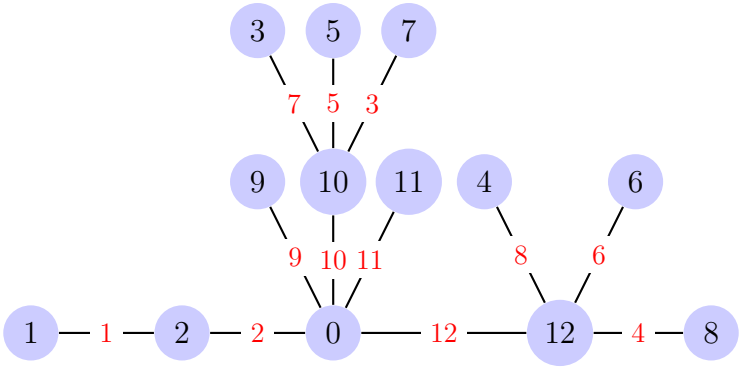


Figure 4.38: A gracefully labeled tree obtained from $T_g^2$ by making several transfers.

The trees obtained in the above figures are lobsters which do not have trivial labelings, and so this expansion procedure can yield interesting trees with corresponding graceful labelings. The power of this procedure comes also from the fact that it can be iterated any number of times to produce trees of greater and greater complexity. For example, a 3-factor

expansion can be performed on the above tree in Figure 4.37, followed by a sequence of transfers of many of the new leaves that would have been attached to the 0 vertex, thus yielding a fairly large and complicated graph that would be nearly impossible to label by hand. It is also important to note the differences between the tree obtained in Figure 4.37 and the tree obtained in Figure 4.38, which highlights that the original labeling of the tree T plays a critical role in determining what trees can be grown from it via this expansion and transfer process.

A shortcoming of this technique is that it is entirely feedforward in nature; that is, we begin with a tree and an explicit graceful labeling of it, and from there we are able to generate trees which are simultaneously labeled as we go. This allows us to create rather complicated graceful trees, but if we instead begin with a large and complex tree that we wish to find a graceful labeling for, it is difficult to imagine how to generate that specific tree using this process. In essence, this would require being able to perform this process in reverse, a possibility we will explore in Section 4.4. For now, we return our attention to the importance of the initial labeling of a tree in determining what can be grown from it via k-factor expansions.

In the current research on graceful labeling, there is a notion of 0-rotatability that we shall define and generalize below.

**Definition 4.10.** Let $G$ be a graceful graph. We say that $G$ is *0-rotatable* if for each $v \in V(G)$, there exists a graceful labeling $f$ such that $f(v) = 0$. We will generalize this notion and say that $G$ is *i-rotatable* if for each $v \in V(G)$, there exists a graceful labeling $f$ such that $f(v) = i$, where $i \in \{0, 1, ..., |V(G)| - 1\}$.

**Remark:** Suppose $G$ is a graceful graph with $n$ edges. We observe that if $G$ is $i$-rotatable then $G$ is also $(n-i)$-rotatable. To see this, note that if $G$ is $i$-rotatable, then for each vertex $v \in V(G)$, there exists a graceful labeling $f$ such that $f(v) = i$, and so $f'(v) = n - i$, where $f'$ is the complementary labeling of $f$.

It is known, for example, that all paths are 0-rotatable (Rosa, [19]). We now extend this

concept of rotatability to vertices and graphs.

**Definition 4.11.** Let $G$ be a graceful graph, and let $v \in V(G)$. We say that $v$ is *vertex rotatable* if for each $i \in \{0, 1, ..., |V(G)| - 1\}$, there exists a graceful labeling $f$ such that $f(v) = i$. Furthermore, we say that $G$ is *fully rotatable* if each vertex in $G$ is vertex rotatable. Equivalently, $G$ is fully rotatable if $G$ is $i$-rotatable for every $i \in \{0, 1, ..., |V(G)| - 1\}$.

**Remark:** A graceful graph $G$ that is 0-rotatable is sometimes said to simply be rotatable. For this reason, graphs that are $i$-rotatable for each $i \in \{0, 1, ..., |V(G)| - 1\}$ are defined as *fully* rotatable above to avoid confusion.

We make these definitions as we expect that the rotatability of a graceful graph will be correlated with the variety of trees that can be generated from it via k-factor expansions and transfers. While this potential relationship is not explored in this paper, we discuss the idea briefly in the section Further Research Ideas. We now demonstrate how this technique of expansion followed by transfers can be utilized to establish some known results.

**Theorem 4.12.** *Let $T$ be a graceful tree with a graceful labeling $f$, and let $v$ be the vertex in $T$ given the maximum label in $f$. Then the tree obtained by attaching $k$ leaves to every vertex other than $v$ in $T$ is graceful. Furthermore, the resulting tree has a graceful labeling where $v$ continues to have the maximum label.*

The above result is one that can be established using a technique known as the $\Delta_{+1}$-construction due to Stanton and Zarnke [20], which is itself a slight modification of the $\Delta$-construction. We define both below.

**Definition 4.13** (Stanton and Zarnke, [20] and Robeva, [18])**.** Let $S$ and $T$ be trees, and let $v$ be a vertex of $T$. Attach one copy of $T$ at each vertex of $S$ by identifying each vertex of $S$ with the vertex corresponding to $v$ in a distinct copy of $T$. We call this construction the $\Delta$-*construction* and denote the resulting tree by $S\Delta T$.

**Definition 4.14** (Stanton and Zarnke, [20] and Robeva, [18])**.** Let $S$ and $T$ be trees, and let $u$ and $v$ be vertices of $S$ and $T$ respectively. Attach one copy of $T$ at each vertex of $S$ other

than $u$ by identifying each vertex of $S$ other than $u$ with the vertex corresponding to $v$ in a distinct copy of $T$. We call this construction the $\Delta_{+1}$-*construction* and denote the resulting tree by $S\Delta_{+1}T$.

With these definitions in mind, we now state the following results without proof.

**Theorem 4.15** (Stanton and Zarnke, [20])**.** *Let $S$ and $T$ be graceful trees. Then $S\Delta T$ is graceful.*

**Theorem 4.16** (Stanton and Zarnke, [20])**.** *Let $S$ and $T$ be graceful trees with graceful labelings $f$ and $g$ respectively. Suppose that $f(u) = |V(S)| - 1$ and $g(v) = 0$. Then $S\Delta_{+1}T$ is graceful.*

We now let $S$ be a graceful tree with graceful labeling $f$, and let $u$ be the vertex in $S$ given the maximum label in $f$. Let $T$ be a star with $k$ leaves and central vertex $v$, and let $g$ be the graceful labeling of $T$ such that $g(v) = 0$. Then $S\Delta_{+1}T$ is a tree precisely of the form obtained in Theorem 4.12, and so such a tree is graceful as desired. The fact that $u$ continues to have the maximum label in the resulting tree can be ascertained by considering the proof in [20]. However, this result can also be determined using the approach of performing a $k$-factor expansion followed by a series of transfers, and we do so now.

*Proof of Theorem 4.12:* Let $T$ be a graceful tree with $n$ vertices and a graceful labeling $f$, and let $v$ be the vertex in $T$ given the maximum label in $f$, i.e. $f(v) = n - 1$. We perform a $(k + 1)$-factor expansion of $T$ given $f$ and note that each vertex $u \in V(T)$ has the label $(k + 1)f(u)$ in $T^f_{k+1}$ and in particular $v$ has the label $(k + 1)(n - 1)$ in $T^f_{k+1}$. Furthermore, we note that the new leaves now attached to the 0 vertex have the labels $1,\ 2,\ ...,\ k,\ (k+1)+1,\ (k+1)+2,\ ...,\ (k+1)+k,\ 2(k+1)+1,\ 2(k+1)+2,\ ...,\ 2(k+1)+k,\ ...,\ (n-2)(k+1)+1,\ (n-2)(k+1)+2,\ ...,\ (n-2)(k+1)+k$, which we will group into the following $n-1$ sets each of order $k$: $\{1, 2, ..., k\},\ \{(k+1)+1, (k+1)+2, ..., (k+1)+k\},\ \{2(k+1)+1, 2(k+1)+2, ..., 2(k+1)+k\},\ ...,\ \{(n-2)(k+1)+1, (n-2)(k+1)+2, ..., (n-2)(k+1)+k\}$.

We will now transfer these sets of leaves to the vertices in $T^f_{k+1}$ corresponding to vertices in $T$ other than $v$ in the following manner. First, we note that we must leave the set with the largest labels i.e. the set $\{(n-2)(k+1)+1, (n-2)(k+1)+2, ..., (n-2)(k+1)+k\}$ attached to the 0 vertex. Of the remaining sets of leaves, we transfer the set with the largest labels to the vertex labeled with $(k+1)$ in $T^f_{k+1}$ i.e. the vertex labeled with 1 in $T$. We then distribute the set with the smallest labels to the vertex labeled with $(k+1)(n-2)$ in $T^f_{k+1}$ i.e. the vertex labeled with $(n-2)$ in $T$. Note that we distributed the largest remaining set to the vertex that had the smallest label other than 0 in $T$, followed by transferring the smallest remaining set to the vertex that had the largest label other than $n-1$ in $T$. We continue transferring these sets of leaves in a similar manner, i.e. we transfer the largest remaining set to the vertex that had the label 2 (the smallest label other than 0 or 1) in $T$, followed by transferring the smallest remaining set to the vertex that had the label $n-3$ (the largest label other than $n-1$ or $n-2$) in $T$, etc. The final result will be a graceful labeling of a graph obtained by attaching exactly $k$ leaves to every vertex in $T$ except for $v$, and where $v$ has the maximum label. □

Below is an example of a simple graceful tree being expanded as described in Theorem 4.12.

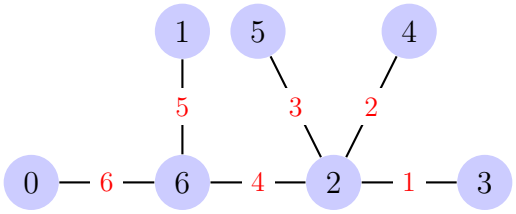


Figure 4.39: A graceful caterpillar $T$ with a graceful labeling $f$.

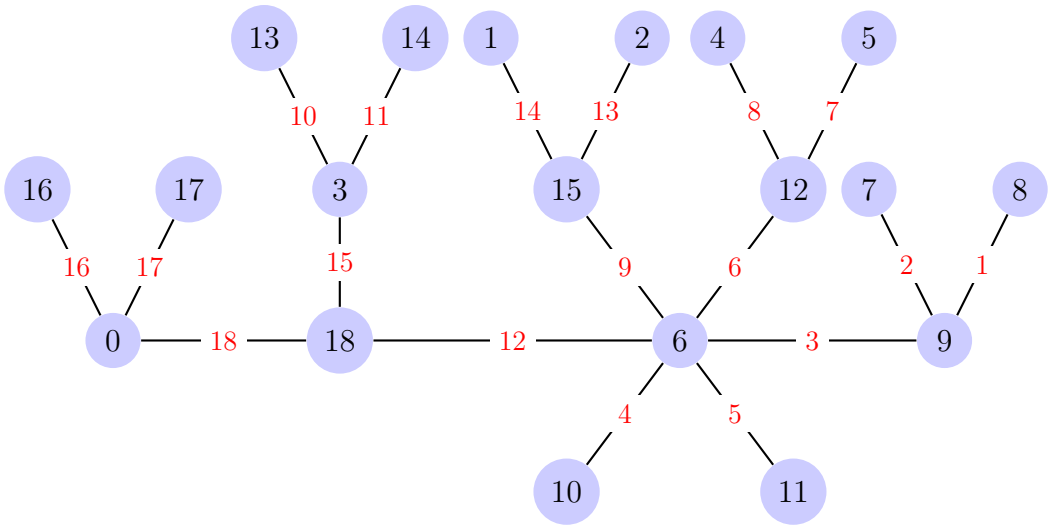


Figure 4.40: The final graceful labeling of the "2-leaf expansion" of $T$.

The following result is now immediate.

**Corollary 4.16.1.** *Let $T$ be a tree where every internal vertex is adjacent to exactly $k$ leaves except for a single internal vertex $v$, where $v$ is not adjacent to any leaves. If the tree obtained by removing these leaves from $T$ is 0-rotatable, then $T$ is graceful. Furthermore, a graceful labeling of $T$ exists such that $v$ is assigned the maximum label.*

## 4.4 A Reductionist Computational Labeling Approach for Trees

A brute force search of all possible labelings, or even a more refined but relatively unsophisticated method as used by Fang, quickly becomes infeasible for finding labelings of large trees, given the exponential growth in complexity of trees as the number of vertices increases. Thus, even though Fang's team was able to prove the gracefulness of all trees with at most 35 vertices [10], their approach might prove impractical for trying to find a graceful labeling for even a single tree of say 60 vertices in a reasonable amount of time. The difficulty in using computational approaches, and indeed with work on the graceful tree conjecture in general, is that it seems to be an entirely global problem. That is, it is generally problematic trying to label individual sections of a tree based on their local structures. Instead, the

entire structure of the tree at large often seems to be significant. For example, adding a single vertex to a previously labeled tree can completely change how it must be labeled to become graceful. I will attempt here to develop a computational approach that in some sense will try and localize the problem, deconstructing a larger tree into a much smaller tree that can then be expanded back into the original tree using the expansion process discussed in Section 4.3. This approach relies on developing heuristics that can be utilized to restrict the search space of potential labelings for the reduced tree to those labelings that are deemed the most well-suited for expanding this reduced tree back into the original tree, preserving a graceful labeling along the way.

Before detailing any heuristics, we first elaborate on the notion of "reducing" a larger tree to a smaller tree. Recall the definition of the $k$-factor expansion procedure defined in Section 4.3 and the observation that if $T$ is a graceful tree with $n$ vertices and a graceful labeling $f$, then its $k$-factor expansion given $f$, $T_f^k$, has $kn-k+1$ vertices. For this reason, a tree $T$ must have $kn-k+1$ vertices for some $k, n \in \mathbb{N}$ in order to perform a $k$-factor reduction. In addition, a $k$-factor expansion is a process that adds leaves to a tree, so we require that a reduction only involve the removal of leaves. In particular, a tree must have at least $(k-1)n-k+1$ leaves in order to perform a $k$-factor reduction, which we now define below.

**Definition 4.17.** Let $T$ be a tree with $kn-k+1$ vertices and at least $(k-1)n-k+1$ leaves for some $k, n \in \mathbb{N}$. We say that a *$k$-factor reduction* of $T$ is a tree $T'$ obtained by removing $(k-1)n-k+1$ leaves from $T$. The set of all possible $k$-factor reductions for $T$ is denoted $T_{(k)}$.

With this definition in hand, our general procedure for searching for graceful labelings will be as follows. Given a tree with $m$ vertices, we will determine what values of $k, n$ satisfy $m = kn-k+1$. Then for each $k, n$ pair, provided that this tree has enough leaves, we consider the possible $k$-factor reductions that can be performed, where the decision of what leaves to remove is partially random but also informed by certain heuristics on how we might label the remaining reduced tree of $n$ vertices. We then find all labelings of this reduced tree that

satisfy these heuristics and see which, if any, of these labelings of this reduced tree can be expanded into a graceful labeling of the original tree via a $k$-factor expansion and subsequent transfers. While not guaranteed to yield a labeling for any given tree, this approach may be able to generate a graceful labeling for certain relatively large trees that would otherwise be difficult or impossible to label. Furthermore, by throwing out the majority of labelings for each reduced tree that do not satisfy our heuristics from the beginning, this allows for a labeling of the original tree to be found in a reasonable amount of time, if a labeling is found at all.

**Remark:** Given a tree $T$, it is possible to perform this procedure on a modified version of $T$ where some leaves have already been removed from a single parent vertex $v$. In this case, we must have that $v$ is assigned the label 0, as then we can trivially add back the initially deleted leaves and give them the highest labels.

Within the scope of my research, I have developed heuristics for performing 2-factor reductions, which can be performed on any tree $T$ with an odd number of vertices and sufficiently many leaves, i.e. a tree with $2k-1$ vertices and at least $k-1$ leaves for some $k$. These efforts were taken to serve as a proof of concept for the merit of this idea and to demonstrate that developing this approach further for general $k$-factor reductions and sequences of reductions is worthwhile. We now explore some heuristics for leaf removal and reduced tree labeling in the 2-factor reduction case.

### 4.4.1 Identifying Candidates for the 0 Label

Given that a 2-factor expansion (and in general a $k$-factor expansion) leaves several leaves attached to the vertex with the 0 label which can then be transferred elsewhere in the tree, it makes sense to assign a vertex with many leaves in the original tree with a label of 0 in the reduced tree. More specifically, we will assign a label of 0 to the vertex from the original tree with the greatest number of leaves removed during the reduction. If multiple vertices have an equal number of leaves removed, one of them will be assigned the label of 0 at random.

### 4.4.2 Odd Label Expansions

When performing a 2-factor reduction on a tree $T$, if an odd number of leaves are removed from a vertex $v$ in $T$ and $v$ is not the vertex being assigned the label of 0, then $v$ must be assigned an odd label in the reduced tree. To see this, we note that any vertex in the reduced tree will have an even label in $T$, including $v$. This is because the reduced tree labels will be multiplied by 2 when performing a 2-factor expansion. Let's say that $v$ has the label $2l$ for some $l$, i.e. $v$ had a label of $l$ in the reduced tree. The leaves of $v$ that were removed in the reduction will be "reattached" by performing transfers from the 0 vertex after making the 2-factor expansion. These transfers will typically be made in pairs, where the labels of the two vertices being transferred sum to $2l$. The one exception is if the 0 vertex has $l$ as a neighbor, in which case we may transfer the single vertex $l$ to $v$. Given that $v$ has an odd number of leaves, such a transfer must occur, but since the leaves we are transferring from 0 all have odd labels, this is only possible if $l$ is an odd number, i.e. $v$ was assigned an odd label in the reduced tree.

### 4.4.3 Even Label Expansions

If an even number of leaves are removed from a vertex $v$ in $T$ during a 2-factor reduction and $v$ is not the vertex being assigned the label of 0, then $v$ may be assigned either an even or odd label in the reduced tree. However, given that vertices that have an odd number of leaves removed must have an odd label in the reduced tree as explained above, we wish to reserve odd labels for such vertices. Therefore, we will require that vertices like $v$ must have an even label in the reduced tree. An alternative approach would be to make the choice of how we label vertices like $v$ probabilistic, i.e. with some (high) probability $p$ we require $v$ to have an even label in the reduced tree, while with probability $1 - p$ we will require it has an odd label. This approach was not explored in this paper, but it is a possibility to consider in future research.

### 4.4.4 A Small Example

We demonstrate the idea behind these heuristics with a small example that can be done without the aid of a computer program. Consider the following tree shown below.

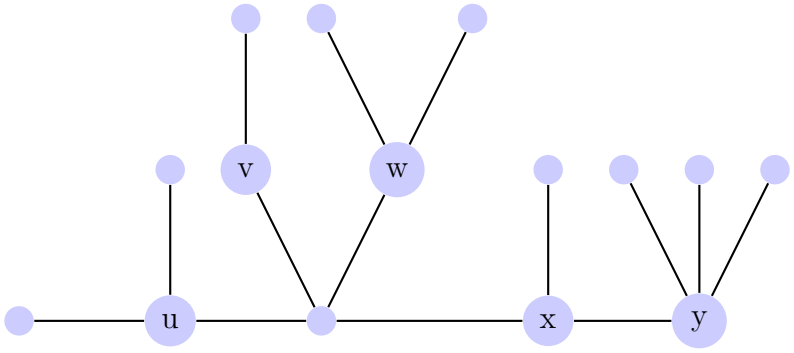


Figure 4.41: A lobster tree $T$.

We note that the above tree $T$ is a lobster and thus it is not initially clear how we might attempt to gracefully label it. However, we also see that $T$ has 15 vertices, where $15 = 2(8) - 1$, and so we may consider performing a 2-factor reduction of $T$ by removing seven leaves from $T$, if possible. In our example, $T$ has nine leaves, and so we may perform such a reduction. We now consider the heuristics in determining which leaves to remove and for what reason in order to inform our choices for how to label their parent vertices.

The vertices that have leaves are $u, v, w, x$, and $y$ as labeled in Figure 4.41. We note that $y$ is a great candidate to be assigned the 0 label, as it has the most leaves. We thus will remove the three leaves from $y$. We would also like to be able to assign $w$ an even label $k$, as then its two leaves can be given odd labels summing to $2k$ after the expansion, and so we will remove the two leaves from $w$. Finally, we note that $v$ and $x$ have exactly one leaf each, and so we can remove those two leaves and seek to assign $v$ and $x$ odd labels. After performing this 2-factor reduction, we are left with the following caterpillar graph, $T' \in T_{(2)}$.

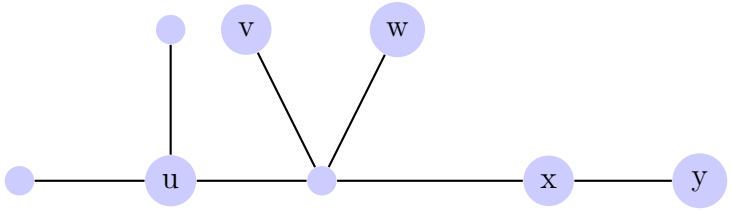


Figure 4.42: A caterpillar graph $T' \in T_{(2)}$, i.e. obtained from performing a 2-factor reduction of $T$.

With our heuristics in mind, we are able to label $T'$ using the straightforward approach for labeling caterpillars, yielding the following labeling, $f$.

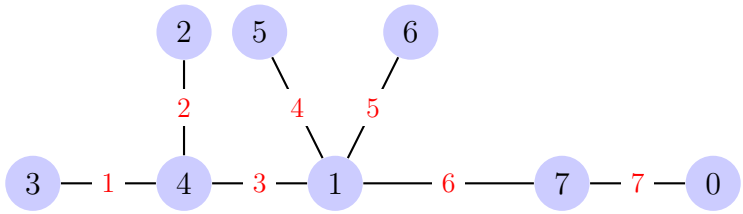


Figure 4.43: A graceful labeling of $T'$ satisfying our heuristics.

We now perform a 2-factor expansion of $T'$ given $f$, as shown below.

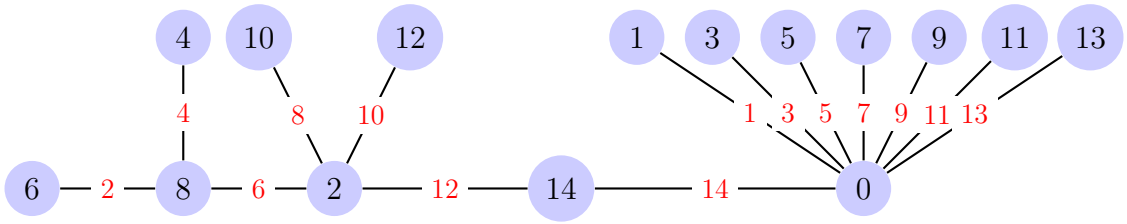


Figure 4.44: $T_f'^2$, the 2-factor expansion of $T'$ given $f$.

Finally, we obtain a graceful labeling of the original lobster T after making the necessary transfers.

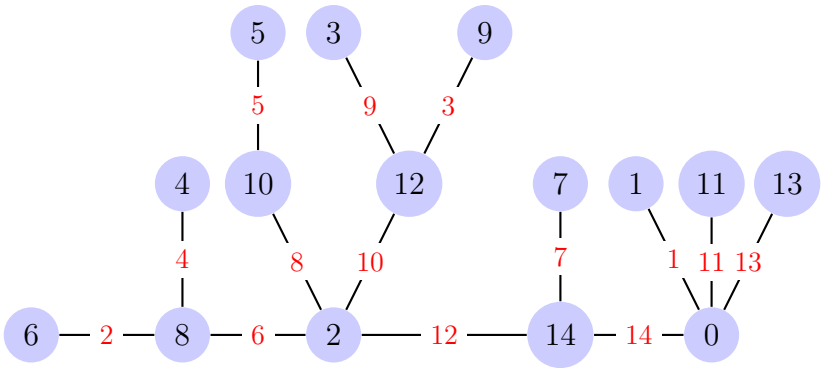


Figure 4.45: A graceful labeling of the lobster tree $T$.

### 4.4.5 The Algorithm

With the above information in mind, we now make our algorithm more explicit. We will represent a tree using the layout representation described in Section 3.3. Given this layout representation, we first determine the adjacency information, i.e. the neighbors of each vertex and which of them are leaves. With this information, we then determine which vertex has the greatest number of leaves. If multiple vertices have the maximum number of leaves, we pick one at random. We remove all of these leaves and record the parent vertex. Of the remaining leaves, we remove the necessary number at random (for a tree with $2k-1$ vertices, this number is $k-1-\#$ of leaves removed from the "maximum" parent vertex), taking note of how many leaves are removed from each parent. We then consider the reduced tree and find its graceful labelings. We remove those labelings which do not match our heuristics to obtain a list of valid "sublabelings."

With these sublabelings in hand, we now attempt to construct a graceful labeling of the original tree. For each parent vertex with an odd number of leaves removed, we first assign one of these leaves the label corresponding to half of the label of the parent vertex. For each of the parent vertices other than the "maximum" parent vertex (which now all have an even number of leaves which need to be labeled) we consider the remaining odd labels available and record the label pairs that are possible. We then make several attempts at randomly assigning the necessary number of valid pairs to each of the parent vertices. If a

successful assignment is made, then a graceful labeling has been achieved and is returned. If no successful assignment is found, then the algorithm was unable to find a graceful labeling using the sublabelings obtained in an earlier step. Given that these sublabelings depend on the initial choice of leaves to remove, which was random, we repeat this process numerous times by making different random selections of which leaves to remove before declaring that we were unable to find a graceful labeling. We include below a couple of examples of trees with graceful labelings that were found using this approach.

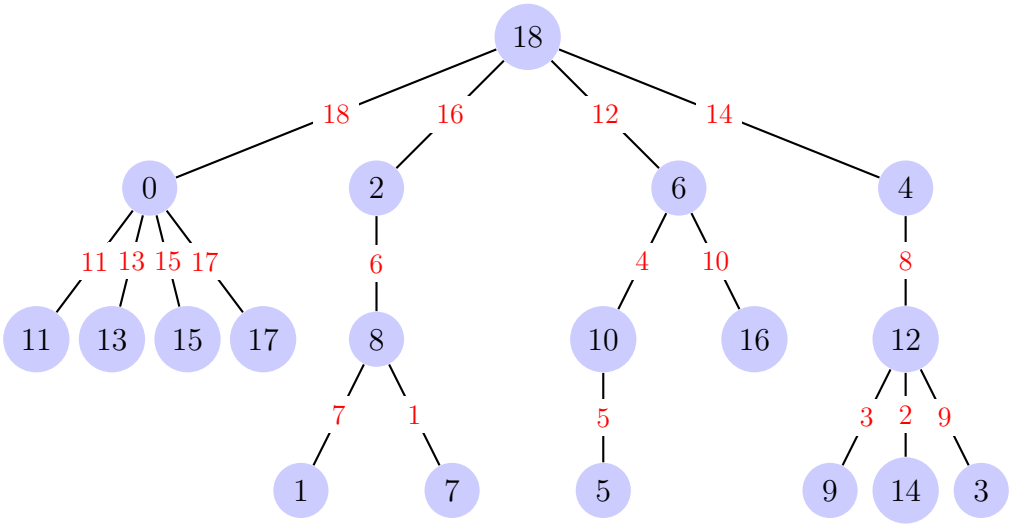


Figure 4.46: A tree $T_1$ with a graceful labeling obtained using the reductionist approach.

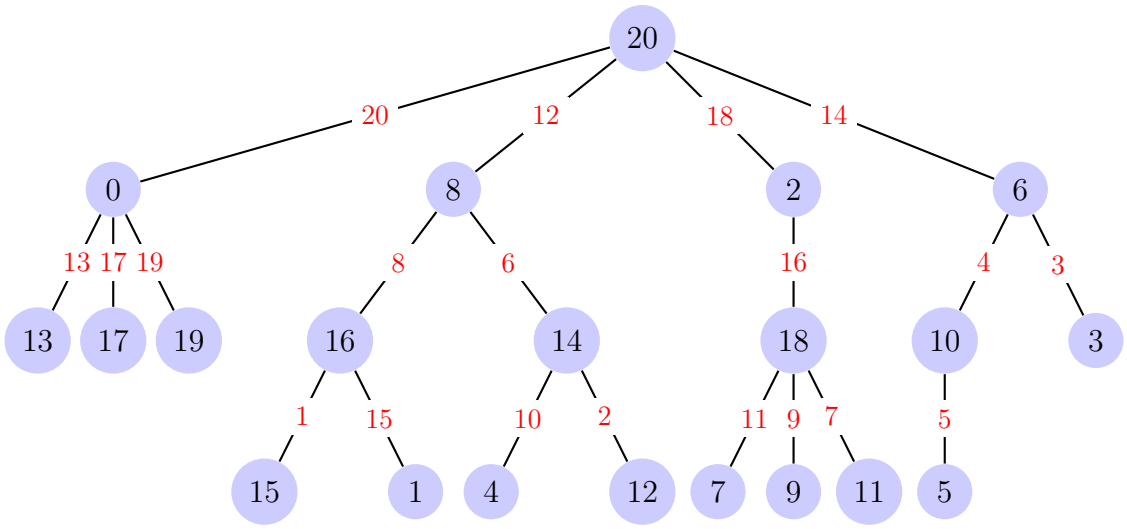


Figure 4.47: A tree $T_2$ with a graceful labeling obtained using the reductionist approach.

# 5 Further Research Ideas

Many of the ideas presented in this paper have the potential to be extended much further. In Section 4.1.1, Proposition 4.3 established which labels can be given to the neighbors of the central vertex of a diameter four tree where the central vertex has degree 2 and is assigned the maximum label. As mentioned previously, exploring the transfers that can be performed on trees obtained using Proposition 4.3 could lead to a complete characterization of diameter four trees with a central vertex of degree 2 such that a graceful labeling exists where the central vertex has the maximum label. Absent from Section 4.1.1 is any discussion of diameter four trees with a central vertex of even degree $d$ with $d \geq 4$, as the analysis becomes more complicated. However, it is my belief that a greater proportion of such trees can be labeled gracefully with the central vertex getting the maximum label due to the greater flexibility that these trees have to offer. More specifically, for any diameter four tree with a central vertex of even degree, we must assign a label of 0 to one of its neighbors. In the case where the central vertex has degree 2, this places immediate restrictions on which labels can be assigned to the second neighbor. When the central vertex has degree at least 4, these restrictions no longer apply as there are additional branches available in the tree to distribute the labels across.

Section 4.1.2 presents some modest progress toward a conjecture by Ghosh [12] and details the types of transfers that are initially possible for some small cases. It is worth investigating whether greater patterns emerge when considering larger cases, in addition to which leaf distributions are possible using these transfers. It should be noted that the transfers detailed in this section are only those transfer pairs that are valid when beginning with balanced lobes. The types of subsequent transfers that can be made after an initial round of transfers has already been made has not been explored, and doing so substantially complicates the analysis. Nevertheless, it may lead to a proof of Ghosh's conjecture or to a proof of the gracefulness of some subset of the lobsters included in the conjecture.

There is much research that can still be done on $k$-factor expansion and transfer sequences. In particular, it is possible that new classes of graceful trees may be established

using this approach. For example, a detailed exploration of performing $k$-factor expansions and transfers on a variety of caterpillar graphs may yield new classes of graceful lobsters. In addition, a careful examination of the types of transfers that can be made after repeated $k$-factor expansions will help inform choices of new heuristics that can expand the reductionist computational approach detailed in Section 4.4 to work for multi-step reductions.

Room for growth exists in the case of single-step reductions as well. As was suggested in Section 4.4.3, it may be worthwhile to consider making the decision on how to label a parent vertex that has had an even number of leaves removed a probabilistic choice. This may allow the algorithm to find new graceful labelings that were previously undetectable using this method. The single-step case can also be expanded to work for general $k$-factor reductions rather than simply 2-factor reductions as it exists now. Such a jump is not too complicated. Whereas leaves are removed one at a time in the case of a 2-factor reduction, the most straightforward approach would be to remove groups of $k-1$ leaves at a time for general $k$-factor reductions, where any given group of $k-1$ leaves is removed from a single parent vertex. Our desired labels for the parent vertices are now analogous to the case where $k = 2$. We would assign a label of 0 to the parent vertex that has the greatest number of groups of leaves removed, an odd label to parent vertices that have an odd number of groups removed, and an even label (in the non-probabilistic case) to parent vertices that have an even number of groups removed. Other transfers become possible when making $k$-factor expansions for $k \geq 3$ – in particular transfers that occur entirely within the leaves themselves – but the ideas presented here are a natural starting point for extending this algorithm.

In Section 4.2 we introduced generalized notions of vertex and graph rotatability and speculate that a relationship may exist between the extent to which a graph is rotatable and the variety of graceful trees that can be expanded from it. It would be interesting to explore whether such a relationship does indeed exist. Furthermore, it may be possible to characterize whether or not a tree is highly rotatable based on local structures within the tree. Additional work might then reveal that those trees which are most rotatable are in fact

the trees that can be labeled successfully using the reductionist approach developed here. This would then allow for a better characterization of those trees which can be labeled with this method. Perhaps enough evidence could even be found to make reasonable a conjecture that all trees with a sufficient number and distribution of leaves relative to the size of the tree are graceful and can be labeled in this manner.

For many of these research efforts, it would be worthwhile to develop an implementation of the tree generation algorithm put forth by Wright et al. [21] and described in Section 3.3. Using this algorithm, trees are represented as layouts detailing the distances of each vertex from the root in the order given by a depth-first traversal of the tree. It is for this exact reason that the layout representation of a tree was my choice for how to represent an input tree to the reductionist algorithm. With a working implementation of this tree generation algorithm, it would then be straightforward to generate trees of a given size and attempt to label each one with the reductionist algorithm rather than having to construct examples individually. It would make efforts such as determining the type and proportion of trees that are highly or fully rotatable easier as well.

# 6 Conclusion

The graceful tree conjecture has been a thorn in the side of graph theorists for decades, and it is likely to remain so for years to come. Still, progress can and has been made on this problem. In this paper some new graceful tree types have been put forth using some well developed techniques for constructing graceful trees. Furthermore, while clever computational approaches for tackling this problem have thus far proven to be elusive, a new approach has been presented that has potential to be expanded to work for large trees that would otherwise be virtually impossible to label gracefully by hand or with current computational methods. The inspiration behind this reductionist algorithm is the notion of graceful tree expansions defined here in this paper. Such expansions serve as a foundation for future graceful tree growth, and understanding how trees may be grown using this procedure will allow for further fine tuning of the reductionist method for generating graceful labelings.

# References


[1] REL Aldred and B.D. McKay. Graceful and harmonious labellings of trees. *Bull. Inst. Combin. Appl*, 23:69-72, 1998.

[2] P. Bahls, S. Lake, and A. Wertheim. Gracefulness of Families of Spiders. *Involve*, 3:241-247, 2010.

[3] J.C Bermond. Graceful graphs, radio antennae and french windmills. *Graph Theory and Combinatorics*, London: Pitman, 18-37, 1979.

[4] J.C. Bermond and D. Sotteau. Graph Decompositions and $G$-design. *Proceedings of the Fifth British Combinatorics Conference*, University of Aberdeen, Aberdeen, 1975.

[5] T. Beyer and S.M. Hedetniemi. Constant time generation of rooted trees. *SIAM Journal on Computing*, 9:706-712, 1980.

[6] G.S. Bloom. A chronology of the Ringel-Kotzig conjecture and the continuing quest to call all trees graceful. *Annals of the New York Academy of Sciences*, 328:32-51, 1979.

[7] L. Brankovic and I. Wanless. Graceful Labelling: State of the Art, Applications and Future Directions. *Mathematics in Computer Science*, 5(1):11-20, 2011.

[8] M. Dinneen. Constant Time Generation of Free Trees. *University of Auckland Lecture*, 2015.

[9] M. Edwards and L. Howard. A survey of graceful trees. *The Altantic Electronic Journal of Mathematics*, 1(1):5-30, 2006.

[10] W. Fang. A computational approach to the graceful tree conjecture. *arXiv:1003.3045 [math.CO]*, 2010.

[11] J.A. Gallian. A dynamic survey of graph labeling. *The Electronic Journal of Combinatorics*, 2015, #DS6.

[12] S. Ghosh. On certain classes of graceful lobsters. *arXiv:1306.2932 [math.CO]*, 2013.

[13] M. Horton. Graceful Trees: Statistics and Algorithms. *Master's Thesis, University of Tasmania*, 2003.

[14] P. Hrnčiar and A. Haviar. All trees of diameter five are graceful. *Discrete Mathematics*, 233:133-150, 2001.

[15] C. Huang, A. Kotzig, and A. Rosa. Further results on tree labellings. *Utilitas Mathematica*, 21:31-48, 1982.

[16] A. Kotzig. On certain vertex-valuations of finite graphs. *Utilitas Mathematica* 4:261-290, 1973.

[17] David Morgan. All lobsters with perfect matchings are graceful. *Electronic Notes in Discrete Mathematics*, 11:503-508, 2002.

[18] E. Robeva. An Extensive Survey of Graceful Trees. *Undergraduate Honors Thesis, Stanford University*, 2011.

[19] A. Rosa. On certain valuations of the vertices of a graph. *Theory of Graphs*, New York: Gordon and Breach, 349-355, 1966.

[20] R.A. Stanton and C.R. Zanke. Labeling of Balanced Trees. *Proceedings of the Fourth South-Eastern Conference on Combinatorics, Graph Theory and Computing*, Boca Raton, FL, 1973.

[21] R.A. Wright, B. Richmond, A. Odlyzko, and B.D. McKay. Constant time generation of free trees. *SIAM Journal on Computing*, 15:540-548, 1986.

# A Appendix

## A.1 Algorithm Code

```
import random
def getAdjacency(L):
    n = len(L)
    adjInfo = []
    for i in range(n):
        neighbors = []
        leaves = []
        d = L[i]
        j = i+1
        while (j < n and L[j] != d):
            if (L[j] == (L[i]+1)):
                neighbors.append(j)
                if (j+1 == n or L[j+1] <= L[j]):
                    leaves.append(j)
            j = j+1
        adjInfo.append((neighbors,leaves))
    return adjInfo


def findSublabelings(A):
    leafCount = 0
    i = 0
    leaves = []
    vertices = []
    edges = []
    maxLeafCount = 0
```

```
maxParents = []
for idx in range(len(A)):
    vertices.append(idx+1)
for t in A:
    for neighbor in t[0]:
        edges.append((i+1,neighbor+1))
    leafCount = leafCount + len(t[1])
    if len(t[1]) > maxLeafCount:
        maxParents = [i+1]
        maxLeafCount = len(t[1])
    elif len(t[1]) == maxLeafCount:
        maxParents.append(i+1)
    for leaf in t[1]:
        leaves.append((leaf+1,i+1))
    i = i+1
subVertices = list(vertices)
subEdges = list(edges)
if leafCount < int(len(A)/2):
    return False
else:
    maxParent = random.sample(maxParents,1)[0]
    maxLeaves = []
    nonMaxLeaves = list(leaves)
    for leaf in leaves:
        if (leaf[1] == maxParent):
            maxLeaves.append(leaf)
            nonMaxLeaves.remove(leaf)
    leavesRemoved = random.sample(nonMaxLeaves,int(len(A)/2)-
```

```
    maxLeafCount)
for ml in maxLeaves:
    leavesRemoved.append(ml)
parentInfo = []
for rl in leavesRemoved:
    subVertices.remove(rl[0])
    subEdges.remove((rl[1],rl[0]))
    piCount = 0
    notIn = True
    for pi in parentInfo:
        if (pi[0] == rl[1]):
            pi0 = pi[0]
            pi1 = pi[1]+1
            pi[2].append(rl[0])
            pi = (pi0,pi1,pi[2])
            parentInfo[piCount] = pi
            notIn = False
            break
        piCount = piCount+1
    if notIn:
        parentInfo.append((rl[1],1,[rl[0]]))
k = 0
for se in subEdges:
    se0 = subVertices.index(se[0])+1
    se1 = subVertices.index(se[1])+1
    se = (se0,se1)
    subEdges[k] = se
    k = k+1
```

```
        subLabelings = graceful(subEdges)
        maxIndex = subVertices.index(maxParent)
        removedLabelings = []
        for labeling in subLabelings:
            if (labeling[maxIndex] != 0):
                removedLabelings.append(labeling)
        for removed in removedLabelings:
            subLabelings.remove(removed)
        newRemovedLabelings = []
        for labeling in subLabelings:
            for p in range(len(parentInfo)-1):
                parentIndex = subVertices.index(parentInfo[p][0])
                leafNumber = parentInfo[p][1]
                if (((labeling[parentIndex] % 2) != (leafNumber %
                    2)) and
                    labeling not in removedLabelings and labeling
                        not in
                    newRemovedLabelings):
                    newRemovedLabelings.append(labeling)
        for newRemoved in newRemovedLabelings:
            subLabelings.remove(newRemoved)
        return (subLabelings, parentInfo, subVertices, edges)


def constructLabeling(A):
    (sublabelings,parentInfo,subVertices,edges) = findSublabelings
        (A)
    print(''     "+str(len(sublabelings))+''valid sublabelings
        found")
```

```
if len(sublabelings) == 0:
    return False
else:
    oddLabels = []
    for i in range(int(len(edges)/2)):
        oddLabels.append(2*i+1)
    for labeling in sublabelings:
        remainingOddLabels = list(oddLabels)
        parentInfoCurrent = []
        for pi in parentInfo:
            pi0 = pi[0]
            pi1 = pi[1]
            pi2 = list(pi[2])
            pi = (pi0,pi1,pi2)
            parentInfoCurrent.append(pi)
        labeling[:] = [2*x for x in labeling]
        fullLabeling = (len(edges)+1)*[1]
        for i in range(len(labeling)):
            fullLabeling[subVertices[i]-1] = labeling[i]
        singleLeaf = []
        for p in range(len(parentInfoCurrent)-1):
            if (parentInfoCurrent[p][1] % 2 == 1):
                fullLabeling[parentInfoCurrent[p][2][0]-1] =
                    int(fullLabeling[parentInfoCurrent[p
                    ][0]-1]/2)
                remainingOddLabels.remove(int(fullLabeling[
                    parentInfoCurrent[p][0]-1]/2))
                parentInfoCurrent[p][2].remove(
```

```
                parentInfoCurrent[p][2][0])
            if len(parentInfoCurrent[p][2]) == 0:
                singleLeaf.append(parentInfoCurrent[p])
            else:
                pi0 = parentInfoCurrent[p][0]
                pi1 = parentInfoCurrent[p][1]-1
                pi = (pi0,pi1,parentInfoCurrent[p][2])
                parentInfoCurrent[p] = pi
    for single in singleLeaf:
        parentInfoCurrent.remove(single)
    allPossiblePairs = []
    for p in range(len(parentInfoCurrent)-1):
        parentLabel = fullLabeling[parentInfoCurrent[p
            ][0]-1]
        possibleLabels = []
        for oddLabel in remainingOddLabels:
            if oddLabel < parentLabel:
                possibleLabels.append(oddLabel)
        possiblePairs = []
        for label in possibleLabels:
            if (parentLabel - label) in possibleLabels:
                if ((label,parentLabel-label) not in
                    possiblePairs
                    and (parentLabel-label,label) not in
                        possiblePairs):
                    possiblePairs.append((label,
                        parentLabel-label))
        allPossiblePairs.append(possiblePairs)
```

```
allPairChoices = []
attempts = 0
while (attempts < 50):
    attempts = attempts + 1
    print(''          Attempt " + str(attempts) + '' to
       assign leaf labels")
    finalLabeling = list(fullLabeling)
    reducedOddLabels = list(remainingOddLabels)
    for p in range(len(parentInfoCurrent)-1):
        if len(allPossiblePairs[p]) < int(
           parentInfoCurrent[p][1]/2):
            return False
        else:
            exitCond = False
            labelsUsed = []
            pairChoices = random.sample(
               allPossiblePairs[p],int(
               parentInfoCurrent[p][1]/2))
            for (p1,p2) in pairChoices:
                if (p1 not in reducedOddLabels or p2
                   not in reducedOddLabels):
                    exitCond = True
                    break
                else:
                    reducedOddLabels.remove(p1)
                    reducedOddLabels.remove(p2)
                    labelsUsed.append(p1)
                    labelsUsed.append(p2)
```

```
                        if exitCond:
                            break
                        for l in range(len(parentInfoCurrent[p
                            ][2])):
                            finalLabeling[parentInfoCurrent[p][2][
                                l]-1] = labelsUsed[l]
                if (len(reducedOddLabels) == len(parentInfoCurrent
                    [len(parentInfoCurrent)-1][2])):
                    for j in range(len(reducedOddLabels)):
                        finalLabeling[parentInfoCurrent[len(
                            parentInfoCurrent)-1][2][j]-1] =
                            reducedOddLabels[j]
                    print("A graceful labeling was found: " + str(
                        finalLabeling))
                    return True
            return False

def labelingSearch(L):
    adjacencyInfo = getAdjacency(L)
    labelingAttempts = 0
    while (labelingAttempts < 25):
        labelingAttempts = labelingAttempts + 1
        print(``Attempt " + str(labelingAttempts) + ``:")
        if (constructLabeling(adjacencyInfo)):
            return
    print(``No graceful labeling was found")
```