Trees, Prüfer Codes, and Parking Functions

Christian Vicars

May 2024

Contents

1 Introduction

In this paper we will present bijections between trees and two different sequences of numbers. In Section 2, we will define spanning trees, leaves, and parking functions to give context for Section 3, where we introduce and prove Cayley's Tree Formula. In Section 3, we will show the map that creates a Prüfer sequence from a labeled tree and vice versa. A Prüfer sequence is a code created by iteratively removing leaves from a tree and recording the labels. We will then show an algorithm that finds a unique parking function from a given Prüfer code. In this way, we will show a map that goes from trees to parking functions, using Prüfer codes as an intermediary step.

2 Definitions and Examples

Definition 1 (Spanning Tree)**.** A spanning tree of connected graph, G, is a connected sub-graph which includes all vertices of G and no cycles.

Example 1. In Figure 1, we see three graphs. The left graph in black is the complete graph with four vertices. Graphs with colors blue and red are examples of spanning trees. As we see, both the blue and red graph connect all vertices but do not contain any cycles. Adding any dashed edge would make either graph no longer a tree.

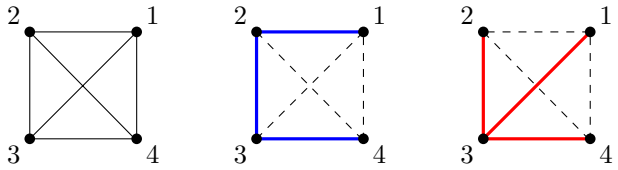

Figure 1: Spanning tree example

Definition 2 (Leaf)**.** A leaf is a vertex of a tree with degree 1

Example 2. In Figure 1, we see the blue tree has two leaves, vertices 1 and 4, and the red tree has three leaves, vertices 1, 2, and 4.

Definition 3 (Parking Function)**.** A parking function is a series of positive integers, $a_1, \ldots, a_n$, such that the increasing rearrangement $b_1, \ldots, b_n$, is such that any $b_i \leq i$.

Example 3 (Parking function example and non-example)**.** The sequence, $\sigma = (3, 2, 5, 6, 1, 1)$ is a parking function. It's weakly increasing rearrangement is $(1, 1, 2, 3, 5, 6)$ and the value of each element is less than or equal to it's index. $\sigma = (4, 3, 2, 5, 2, 3)$ is not a parking function. Its weakly increasing rearrangement is $b = (2, 2, 3, 3, 4, 5)$, and clearly in the first index $b_1 \nleq 1$ as $2 \nleq 1$. Thus is is not a parking function.

Now that we have definitions for trees and leaves, we will show the process of creating a Prüfer sequence from a labeled tree and vice versa. By creating a bijective map between the two, we are easily able to count the number of Prüfer sequences which also gives us a count of spanning trees.

3 Cayley's Tree Formula

Theorem 1 (Cayley's Tree Formula)**.** *For any positive integer n, the number of trees with n labeled vertices is n^{n-2}*

Note that Cayley's Tree Formula also counts the number of spanning trees of the complete graph, K_n.

To prove Cayley's Formula, we will show there exists a bijection by defining a map from sequences of length $n-2$ who's entries are $\{0, 1, \ldots, n-1\}$, and trees with n vertices. This map was found by Heinz Prüfer, thus we call these sequences Prüfer sequences or Prüfer codes. By showing the map is an inverse of itself, we will have shown the number of trees with n labeled vertices and Prüfer sequences are the same, and thus be able to count the number of Prüfer sequences to prove Cayley's Tree Formula.

Proof. We will first show the process of creating a sequence from a tree. Consider a tree with n vertices, labeled from $\{0, 1, \ldots, n-1\}$. To create a sequence from this tree of length $n-2$, we will remove leaves until two vertices remain. Generally, at the $i-th$ step, remove the leaf with the smallest label, and record the label of it's neighbor for the Prüfer sequence. An example is shown in Figure 2.

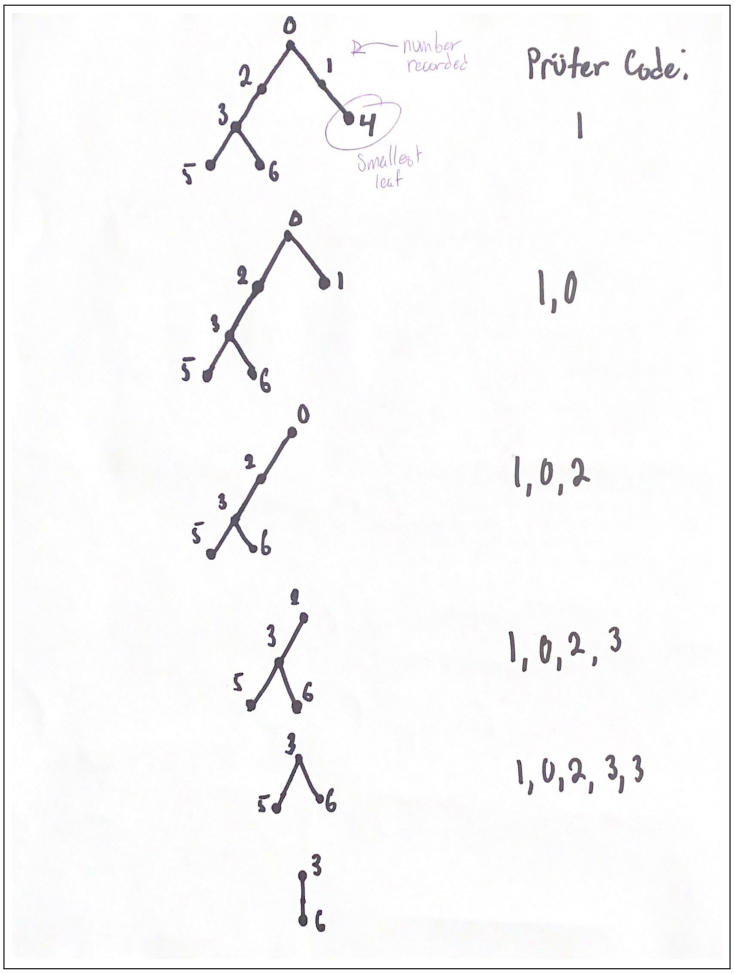

Figure 2: Finding a Prüfer sequence from a labeled tree

We will now show the process of creating a tree from a sequence. Consider a sequence of length $n-2$, who's entries are from $\{0, 1, \ldots, n\}$. Draw n vertices and label them $v_0, v_1 \ldots, v_n$. Denote $S = \{0, 1, \ldots, n\}$. Let j be the smallest number in S that does not appear in our sequence. Place an edge between v_j and the vertex whose subscript appears first in our sequence. Next, remove the first number from our sequence and remove the element j from S. Continue

this process until our sequence is empty, and place an edge between the vertices who's subscripts are left in S. By using set S, we create exactly $n-1$ connected edges, thus ensuring we create a tree. An example is shown in Figure 3.

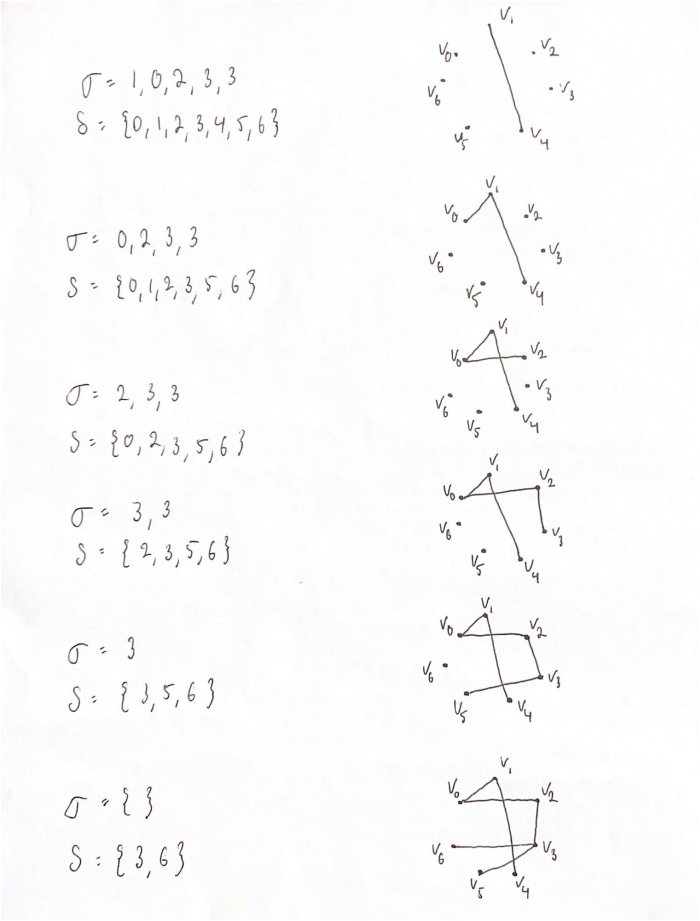

Figure 3: Finding a labeled tree with a Prüfer code

Due to the bijection between labeled trees with n vertices and sequences of length $n-2$ who's entries are in $\{0, 1, \ldots, n-1\}$, we count the number of sequences as n^{n-2} where we have $n-2$ decision to pick from n numbers. Thus we have also counted labeled trees with n vertices as n^{n-2}, proving Cayley's Theorem. □

4 Prüfer Code to Parking Function Algorithm

We will now show a bijection between spanning trees on $n+1$ labeled vertices to parking functions of length n exists. The following theorem from Cather-

ine Yan's, *Handbook of Enumerative Combinatorics*, provides a relationship between parking functions and Prufer sequences As we will see in the following algorithm in Section 4, there is a very simple way to create Prüfer codes from parking functions, however finding the parking function from a Prüfer code is a bit more involved.

Note that we are now using spanning trees on $n+1$ vertices instead of n, as well as the fact that parking functions have one more entry than Prüfer codes. Also note that the term Prüfer codes and difference sequence are synonymous.

Theorem 2. *For any parking function $a = (a_1, \ldots, a_n)$, we define the difference sequence $(c_1, \ldots c_{n-1})$ such that*

$$c_i \equiv a_{i+1} - a_i \mod n+1 \tag{2.1}$$

Corollary 1 (Inverse of Theorem 2)**.**

$$a_i \equiv a_1 + c_1 + \cdots + c_{i-1} \mod n+1.$$

Notice how we can find a parking function directly from a Prüfer sequence so long as we know the value for a_1. The following algorithm is used to find a_1 from a given Prüfer sequence, $(c_1, \ldots c_{n-1}) \in \{0, 1, \ldots, n\}^{n-1}$. We will have an ongoing example to illustrate each step.

Algorithm to find a_1: Given $(c_1, \ldots, c_n) \in \{0, 1, \ldots, n\}^{n-1}$

Step ①:

$$\begin{aligned} h_1 &= 0 \\ h_i &= c_1 + \cdots + c_{i+1} \mod n+1, \qquad 2 \le i \le n \end{aligned}$$

Step ① Example: Suppose we have difference sequence $c = (1, 0, 2, 3, 3) \in \{0, 1, \ldots, 6\}^5$, where $n = 6$. Thus, using ①, we have,

$$\begin{aligned} h_1 &= 0 & h_4 &= 1+0+2 = 3 \\ h_2 &= 1 & h_5 &= 1+0+2+3 = 6 \\ h_3 &= 1+0 = 1 & h_6 &= 1+0+2+3+3 = 9 \equiv 2 \mod 7 \end{aligned}$$

thus $h = (0, 1, 1, 3, 6, 2)$.

Step ②: Define $r(h) = (r_0, \ldots, r_n)$ to be the *specification* of h, such that $r_i = |j : h_j = i|$. In other words, each r_i is equal to the number of times i shows up as a value in h.

We will also define $R_j(h) = r_0 + \cdots + r_j - j - 1$ for $0 \leq j \leq n$.

Step ② Example: With $h = (0, 1, 1, 3, 6, 2)$, we have,

$$\begin{aligned} r_0 &= 1 & r_4 &= 0 \\ r_1 &= 2 & r_5 &= 0 \\ r_2 &= 1 & r_6 &= 1 \end{aligned}$$

thus $r(h) = (1, 2, 1, 0, 0, 1)$ (that is, the value 0 shows up once in h, 1 shows up twice, ect.).
Now we find $R(h)$, such that

$$\begin{aligned} R_0 &= 1 - 0 - 1 = 0 & R_4 &= 1 + 2 + 1 + 1 + 0 - 4 - 1 = 0 \\ R_1 &= 1 + 2 - 1 - 1 = 1 & R_5 &= 1 + 2 + 1 + 1 + 0 + 0 - 5 - 1 = -1 \\ R_2 &= 1 + 2 + 1 - 2 - 1 = 1 & R_6 &= 1 + 2 + 1 + 1 + 0 + 0 + 1 - 6 - 1 = -1 \\ R_3 &= 1 + 2 + 1 + 1 - 3 - 1 = 1 \end{aligned}$$

thus $R(h) = (0, 1, 1, 1, 0, -1, -1)$.

Step ③: Define d to be the smallest index such that $R_d(h) = min\{R_j(h) : 0 \leq j \leq n\}$. In other words find the smallest index where the smallest number in $R(h)$ shows up. Then we have

$$a_1 = n - d$$

Step ③ Example: For $R(h) = (0, 1, 1, 1, 0, -1, -1)$, the smallest number to appear is -1 and this appears in index 5. Thus, $d = 5$, and thus $a_1 = 6 - 5 = 1$. Knowing our a_1 value, we can now find our parking function using the inverse equation from Theorem 2. Using the steps as above, for Prüfer sequence $c = (1, 0, 2, 3, 3)$, our parking function is $(1, 2, 2, 4, 0, 3)$. This concludes the algorithm for finding a_1 from a Prüfer code.

To find a Prüfer code from a given parking function, we simply use Corollary 1. From our parking function $(0, 1, 1, 3, 6, 2)$, we subtract each value from the one before modulo $n+1$, starting on the second index, providing us with Prüfer code, $(1, 0, 2, 3, 3)$.

This map has shown a bijection between the number of parking functions with n numbers and the number of trees on $n + 1$ labeled vertices. Using Cayley's Tree Formula, the number of trees with $n+1$ vertices is $(n+1)^{(n+1)-2} = (n + 1)^{n-1}$, thus, this is also equal to the number of parking function with n entries .

5 Conclusion

In this paper we demonstrated three things that are counted by $(n+1)^{n-1}$, namely, parking functions and labeled trees using bijections, as well as Prüfer codes used as an intermediary step. While we showed one proof of Cayley's Tree Formula in this paper, the curious reader can find a short and tidy proof using Kirchhoff's matrix theorem on Wikipedia. Cayley's Tree Formula gives a count of rooted forests, which are rooted trees allowing disconnection. These rooted forests are counted identically to parking function. In addition, a particular type of tree called binary trees can be counted by the Catalan numbers.

References

[1] Catherine H. Yan. Parking functions in *Handbook of enumerative combinatorics*

[2] John M. Harris Jeffery, L. Hirst, and Michael J. Mossinghoff. *Combinatorics and Graph Theory*, Second Edition, Springer.