Parking Functions and Labeled Trees : Inversions of Labeled Trees

Janitha Aswedige

Abstract

This is a report on the sections 1.1 and 1.2 of the book "Handbook of Enumerative Combinatorics". We fill in the details of the proofs and construct several examples to understand the content better.

Contents

1 Introduction to Parking Functions

Suppose we have n cars $C_1, \ldots, C_n$. They want to park on a one-way street with ordered parking spaces $0, \ldots, n-1$. Each car has a preferred parking space a_i. The card enter the street one at a time in the order $C_1, \ldots, C_n$. Each car tried to park in its preferred space. If the preferred space is taken, it parks in the next available space. If no space is left, then the car leaves the street. The sequence $\mathbf{a} = (a_1, \ldots, a_n)$ is called a parking function of length n if all the cars can park.

Assume we have only three cars C_1, C_2, C_3, and $a_1 = 1,\ a_2 = 2$, and $a_3 = 2$. Then $(a_1, a_2, a_3) = (1, 2, 2)$ is not a parking function as C_3's preferred spot is 2, and it's taken by C_2, and there are no spot to the right of 2. Some examples of parking functions are $(0, 1, 2)$ and $(0, 1, 1)$.

We'll characterize parking functions in the following lemmas. But before that we'll look at some concrete examples.

Suppose we have 4 cars C_1, C_2, C_3, and C_4. And they need to park in the parking spaces $a_1, a_2, a_3, a_4 \in \{0, 1, 2, 3\}$. Assume $\mathbf{a} = (a_1, a_2, a_3, a_4)$ is a parking function. Then at least one of a_i's must be a 0. Because otherwise the four cars will have to park in at most 3 parking spots which is absurd by the pigeonhole principle. Likewise there must be at least two a_i's which are less than 2. If not, there will be exactly one 0 and three a_i's greater than or equal to 2, that is there will be exactly one car which wants to park at the space 0, and three cars which want to park at a space greater than or equal to 2. This is also impossible by the pigeonhole principle. So there must be at least two a_i's less than 2. Similarly we see that there must be at least three a_i's less than 3. And clearly all the a_i's must be less than 4. So there are at least four a_i's less than 4.

On the other hand if there are at least k a_i's less than k for each $k = 1, 2, 3, 4$ we see that each car can park without leaving the street. The reason for this is the following. If there is a conflict of preferences then the cars can always resolve in to agreeing to park in some combination of $(0, 1, 2, 3)$, which indeed allows all of them to park without leaving the street.

Generalizing the above fact we immediately have the following lemma.

Lemma 1.1. $\mathbf{a} = (a_1, \ldots, a_n)$ *is a parking function if and only if* $\mathbf{a}$ *has at least* i *terms less than* i *for each* $1 \leq i \leq n$.

Lemma 1.2. *Let* $a_{(1)} \leq a_{(2)} \leq \cdots \leq a_{(n)}$ *be the non-decreasing rearrangement of* $\mathbf{a} = (a_1, \ldots, a_n)$. *Then* $\mathbf{a}$ *is a parking function if and only if* $0 \leq a_{(i)} < i$.

Proof. This follows immediately from Lemma 1.1 and the fact that each a_i is an integer. For example $0 \leq a_{(1)} < 1$ forces $\mathbf{a}$ to have at least one 0, and $0 \leq a_{(2)} < 2$ forces $\mathbf{a}$ to have at least two terms less than 2, and so on. □

Lemma 1.3. *Let* $\mathbf{a} = (a_1, \ldots, a_n)$. *Then* $\mathbf{a}$ *is a parking function if and only if there exists* $\sigma \in S_n$ *so that* $0 \leq a_{\sigma(i)} < i$.

Proof. This follows from Lemma 1.1 and Lemma 1.2. □

Next we'll count the number of parking functions of length n.

Theorem 1.4. *The number of parking functions of length* n *is* $(n+1)^{n-1}$.

Proof. This proof is due to Pollak. Add an extra space $n+1$ and arrange $0, 1, \ldots, n+1$ in a circle clockwise. Preferences a_i are in $\{0, 1, \ldots, n\}$. So we have $(n+1)^n$ sequences $(a_1, \ldots, a_n)$. We treat $a_i = n$ as any other preference. If space n is occupied C_i moves clockwise to the first unoccupied space. Observe that any sequence of preferences leaves one space unoccupied. By symmetry the number of sequences leaving the space k is the same for all $0 \leq k \leq n$. But if $k = n$, then we get a parking function. Combining the last two sentences we get the number of parking functions is $\frac{1}{n+1}$ of the total number of sequences, which gives

$$\text{Number of parking functions } = \frac{1}{n+1}(n+1)^n = (n+1)^{n-1}.$$

□

2 Some Notation

We write $[n]$ for the set $\{1, 2, \ldots, n\}$ and $[n]_0$ for the set $\{0, 1, \ldots, n\}$. A function $f : [n] \to [n]$ is viewed as the sequence $f(1), f(2), \ldots, f(n)$. If (h_i) is a sequence denote it by the boldface letter $\mathbf{h}$. The boldface letter $\mathbf{a}$ is reserved for a parking function of length n. The set of all parking functions of length n is denoted by $\mathcal{PK}_n$. By Theorem 1.4 we have $|\mathcal{PK}_n| = (n+1)^{n-1}$.

A labeled tree on $[n]_0$ is a connected graph on the vertex set $[n]_0$ with no cycles. The set of all labeled trees with vertex set $[n]_0$ is denoted by $\mathcal{T}_{n+1}$.

3 Labeled Trees with Prüfer Code

A Prüfer code associates with each labeled tree with n vertices a unique sequence of length $n-2$ via the following algorithm.

Step 1 : Remove the leaf with smallest label and set

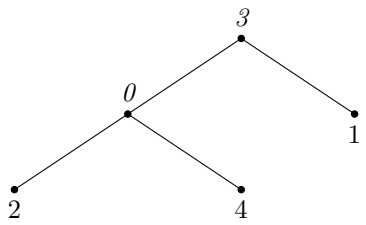
$$c_1 = \text{ label of the neighbouring leaf of the the removed leaf.}$$

Step i : Remove the leaf with smallest label and set

$$c_i = \text{ label of the neighbouring leaf of the the removed leaf.}$$

Let's look at an example.

Example 3.1. *Consider the following tree.*



It has the Prüfer code $(3, 0, 0)$. *Note that once the algorithm ends we're left with the edge connecting* 0 *and* 4. *This is always the case; we'll be left with a single edge connecting two nodes. So the length of the Prüfer code for a labeled tree with* n *vertices is indeed* $n - 2$.

For each parking function of length n we can associate a difference sequence $(c_1, \ldots, c_{n-1})$ of length $n-1$ defined as follows.

$$\begin{aligned} c_1 &= a_2 - a_1 && \mod n+1 \\ &\vdots \\ c_{n-1} &= a_n - a_{n-1} && \mod n+1 \end{aligned}$$

By Theorem 1.4 and Cayley's formula we have $|\mathcal{PK}_n| = |\mathcal{T}_{n+1}|$. It turns out that there is an explicit bijection between $\mathcal{PK}_n$ and $\mathcal{T}_{n+1}$ which involves Prüfer codes. The following result is due to Pollack.

Theorem 3.1. *For each parking function* $\mathbf{a} = (a_1, \ldots, a_n)$, *define the difference sequence* $(c_1, \ldots, c_{n-1})$ *as above. Let* $T(\mathbf{a})$ *be the labeled tree in* $\mathcal{T}_{n+1}$ *whose Prüfer code is* $(c_1, \ldots, c_{n-1})$. *Then the map* $\mathbf{a} \mapsto T(\mathbf{a})$ *is a bijection from* $\mathcal{PK}_n$ *to* $([n_0])^{n-1}$.

Proof. Since $c_1 = a_2 - a_1 \mod n+1$ we have $a_2 = c_1 + a_1 \mod n+1$. Since $c_2 = a_3 - a_2 \mod n+1$ we have $a_3 = c_2 + a_2 \mod n+1 = a_1 + c_1 + c_2 \mod n+1$. Continuing in this manner we have

$$a_i = a_1 + c_1 + \cdots + c_{i-1} \mod n+1 \text{ for all } 2 \leq i \leq n. \tag{1}$$

So we see that the difference sequence $(c_1, \ldots, c_{n-1})$ determines the parking function $\mathbf{a}$ if we know a_1. So our goal is to determine an algorithm to find a_1 for each $(c_1, \ldots, c_n) \in \{0, 1, \ldots, n\}^{n-1}$ such that the sequence $(a_1, \ldots, a_n)$ determined by Equation (1) is a parking function.

Algorithm to determine a_1: Given $(c_1, \ldots, c_{n-1}) \in [n_0]^{n-1}$.

- Step 1 : Let $h_1 = 0$, $h_i = c_1 + \cdots + c_{i-1} \mod n+1$ for $2 \leq i \leq n$.
- Step 2 : Let $\mathbf{r}(h) = (r_0, \ldots, r_n)$ where $r_i = |\{j : h_j = i\}|$.[1] Let

  $$R_j(h) = r_0 + \cdots + r_j - j - 1$$

  for $0 \leq j \leq n$. Here h is the sequence $(h_1, \ldots, h_n)$.
- Step 3 : Let d be the smallest index so that

  $$R_d(h) = \min\{R_j(h) : 0 \leq j \leq n\},$$

  and put

  $$a_1 = n - d.$$

□

Let's look at some examples.

[1] $\mathbf{r}(h)$ is called the *specification* of h.

Example 3.2. *Prüfer code is* $(2,4,7,0,2,5)$. *Here* $n=7$. *First we'll determine the sequence* h. h_1 *is always* 0. *And* $h_1=2,\ h_3=6,\ h_4=5,\ h_5=5,\ h_6=7,$ *and* $h_7=4$. *So* $h=(0,2,6,5,5,7,4)$. *Next we determine* $\mathbf{r}(h)$, *the specification of* h. *Note that* r_i *gives the number of cars whose preferred spot is* i *according to* h. *We'll say more about this in our next examples. For now, we'll just follow the algorithm. So we have* $\mathbf{r}(h)=(1,0,1,0,1,2,1,1)$. *The final step is to determine* $R(h)$. $R(h)$ *is indeed the sequence* $(0,-1,-1,-2,-2,-1,-1,-1)$. *And the minimum value of* $R(h)$ *occurs at the third index, i.e.* $d=3$. *Therefore* $a_1=n-d=7-3=4$. *So we get the parking function* $\mathbf{a}=(4,6,2,1,1,3,0)$.

We observed that the sequence h is an attempt at a parking function. If we consider the entries in h as the preferences of the cars $C_1,\ldots,C_n$, then what the sequence h does is allowing the cars to park in a circular street which has $n+1$ parking spaces. In doing so, there will be a vacant parking spot. And the rest of the algorithm determines the number of clockwise rotations the cars need to perform so that the $(n+1)$-th parking space is empty, thus allowing all the cars to park as specified in the original parking problem. This is essentially Pollack's idea in his counting parking functions proof. So in case h allows the cars to park in the first n parking spots with zero cars leaving the street, that is with $(n+1)$-th spot vacant, then we can always stop at h. Because once we calculate the sequence $R(h)$ we'll see that $d=n$ and therefore $a_1=n-n=0$. This scenario is illustrated in the following examples.

Example 3.3. *Prüfer code is* $(1,3,7,0,2,4)$. *Here* $n=7$. *We can check that the sequence* h *is* $(0,1,4,3,3,5,1)$. *We'll eventually see that* h *is the required parking function of length* 7. *And the specification of* h *is* $\mathbf{r}(h)=(1,2,0,2,1,1,0,0)$. *The sequence* $R(h)$ *is* $(0,1,0,1,1,1,0,-1)$. *And* -1 *is the minimum and it occurs at* $d=7$. *So* $a_1=n-d=7-7=0$. *That is,* h *is indeed the parking function output by the algorithm.*

4 Inversions of Labeled Trees

Start with a parking function $\mathbf{a}=(a_1,\ldots,a_n)$ of length n. We've already seen that not every car gets to park in it's preferred parking spot. So assume C_i parks at p_i for each $1\le i\le n$. The meaning of p_i is the following. If a_i is unoccupied and it's C_i's turn to park then $p_i=a_i$. And if a_i is occupied and it's C_i's turn, then $p_i\neq a_i$, and indeed $p_i>a_i$. But at the end of the day $p_1,\ldots,p_n$ is always going to be a permutation on $\{0,1,\ldots,n-1\}$.

Definition 4.1. *Let* $\mathbf{a}=(a_1,\ldots,a_n)$ *be a parking function of length* n. *The total displacement, denoted by* $D(\mathbf{a})$, *is defined as follows.*

$$D(\mathbf{a})=\sum_1^n(p_i-a_i)=\binom{n}{2}-\sum_1^n a_i. \tag{2}$$

Total displacement $D(\mathbf{a})$ is the total number of *failed trials* before each car find its parking space. Hence it's also called the *total inconvenience*. Let's look at some examples.

Example 4.1. $n=4$ *and* $\mathbf{a}=(0,0,0,0)$. *Then* $p_1=0,\ p_2=1,\ p_3=2,$ *and* $p_4=3$. *Moreover,* $\sum_1^4 a_i=0$. *Thus*

$$D(\mathbf{a})=\sum_1^4(p_1-a_i)=6=\binom{4}{2}-\sum_1^4 a_i.$$

Example 4.2. *For the parking function* $\mathbf{a}=(1,0,3,2)$ *the total displacement is* 0, *because there is no inconvenience experienced by the cars when finding their parking spots. Equation* (2) *indeed says so. It's simply because* $p_i=a_i$ *for each* i, *and* $\binom{4}{2}=6=\sum_1^4 a_i$.

Definition 4.2. *The displacement enumerator of parking functions is the polynomial*

$$P_n(q)=\sum_{\mathbf{a}\in\mathcal{PK}_n}q^{D(\mathbf{a})}=\sum_{\mathbf{a}\in\mathcal{PK}_n}q^{\binom{n}{2}-(a_1+\cdots+a_n)}=q^{\binom{n}{2}}\sum_{\mathbf{a}\in\mathcal{PK}_n}q^{-(a_1+\cdots+a_n)}. \tag{3}$$

The degree of $P_n(q)$ is defined to be $\binom{n}{2}$.

We'll look at an example.

Example 4.3. *We know that $\mathcal{PK}_2 = \{(0,0),(0,1),(1,0)\}$. A nonzero total inconvenience/displacement occurs only in $(0,0)$. Thus*

$$P_2(q) = q^1 + q^0 + q^0 = q + 2.$$

This is also satisfied by each of the equalities in Equation (3). And we see that the degree of $P_2(q)$ is indeed $1 = \binom{2}{2}$.

Next we'll look at inversions of trees. What follows is a few definitions. Recall that $\mathcal{T}_{n+1}$ is the set of all labeled trees with vertex set $[n]_0$. We'll view 0 as the root of T for $T \in \mathcal{T}_{n+1}$. Let $T \in \mathcal{T}_{n+1}$. An *inversion* in T is a pair of vertices labeled i, j such that $i > j$ and i is on the unique path from 0 to j in T.

Definition 4.3. *Let $T \in \mathcal{T}_{n+1}$. Then $inv(T)$ = Number of inversions in T.*

Recall that an inversion in a permutation $\sigma_1, \ldots, \sigma_{n+1}$ of $\{0, \ldots, n\}$ is a pair (σ_i, σ_j) so that $i < j$ and $\sigma_i > \sigma_j$. It's clear that the notions of inversions in permutations and inversions in trees coincide if the tree is a path.

Example 4.4. *There are three non-isomorphic labeled trees in $\mathcal{T}_2$. The only tree in $\mathcal{T}_2$ with an inversion is the following path. And it has exactly one inversion.*

Definition 4.4. *The inversion enumerator $I_n(q)$ of labeled trees on $n+1$ vertices, i.e. with the vertex set $[n]_0$, is the polynomial*

$$I_n(q) = \sum_{T \in \mathcal{T}_{n+1}} q^{inv(T)}. \tag{4}$$

Example 4.5. *Look at $\mathcal{T}_2$. Then in Example 4.4 we saw there are three non-isomorphic labeled trees and exactly one of them has an inversion. So we have $I_2(q) = q^0 + q^0 + q^1 = 2 + q$. Here's where things get prettier! Compare the polynomial $I_2(q)$ with $P_2(q)$ in Example 4.3. They are the same! That is,*

$$I_2(q) = 2 + q = P_2(q).$$

It turns out that this isn't merely a fluke.

Theorem 4.1. *We force $I_0(q) = 1 = P_0(q)$. We have the following recurrence relations.*

1. *(a) $I_1(q) = 1$,*
   (b) $I_{n+1}(q) = \sum_{i=0}^{n} \binom{n}{i}(q^i + q^{i-1} + \cdots + 1) I_i(q) I_{n-i}(q)$.
2. *(a) $P_1(q) = 1$,*
   (b) $P_{n+1}(q) = \sum_{i=0}^{n} \binom{n}{i}(q^i + q^{i-1} + \cdots + 1) P_i(q) P_{n-i}(q)$.
3. *Consequently, $I_n(q) = P_n(q)$.*

We will not discuss the proof of this theorem. However it's noticed that item 3 follows immediately from items 1 and 2, and by an induction on n. Moreover, we notice that $P_n(1)$ gives the number of parking functions of length n. We'll in fact attempt to prove this.

Theorem 4.2. *For each n we have $P_n(1) = (n+1)^{n-1}$.*

Next we describe a bijection $\phi : \mathcal{PK}_n \to \mathcal{T}_{n+1}$ so that $D(\mathbf{a}) = inv(\phi(\mathbf{a}))$, which is due to Knuth. We'll describe the algorithm using an example.

Let $\mathbf{a} = (a_1, \ldots, a_n) \in \mathcal{PK}_n$ and p_i be the space C_i occupies. Put $p'_i = p_i + 1$ and $\mathbf{p}' = p'_1 \ldots p'_n$. Then $\mathbf{p}'$ is a permutation of length n. Let $\mathbf{q} = q_1 \ldots q_n$ be the inverse of $\mathbf{p}'$. This means C_{q_i} is parked at $i - 1$.

Take $\mathbf{a} = (4, 0, 1, 0, 3, 6, 4) \in \mathcal{PK}_7$. Then the cars $C_1, \ldots, C_7$ are parked as follows.

$$C_2\ C_3\ C_4\ C_5\ C_1\ C_7\ C_6$$

And we have

$$\mathbf{p} = (4, 0, 1, 2, 3, 6, 5)$$

and

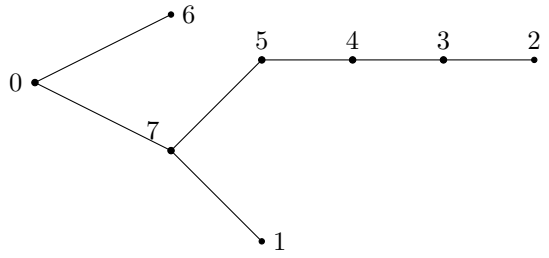
$$\mathbf{p}' = (5, 1, 2, 3, 4, 7, 6).$$

Note that $\mathbf{p}'$ in cycle notation is (1 5 4 3 2)(6 7). So its inverse $\mathbf{q}$ in cycle notation is is (5 1 2 3 4)(6 7). And therefore $\mathbf{q} = (2, 3, 4, 5, 1, 7, 6)$.

Step 1: Construct an auxiliary tree as follows.

Let the predecessor of vertex k be the first element on the right of k and larger than k in $\mathbf{q}$. If no such element exists, then let the predecessor be 0.

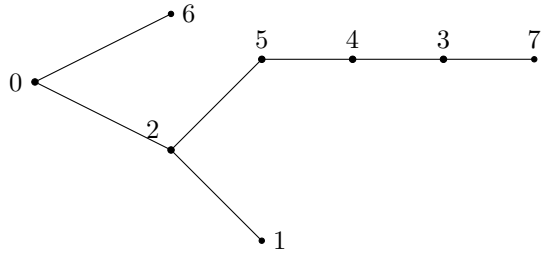
We'll denote *"i is the predecessor of j"* by $i \prec j$. Then we have $7 \prec 1, 0 \prec 7, 3 \prec 2, 7 \prec 5 \prec 4 \prec 3$, and $0 \prec 6$. So we get the auxiliary tree :



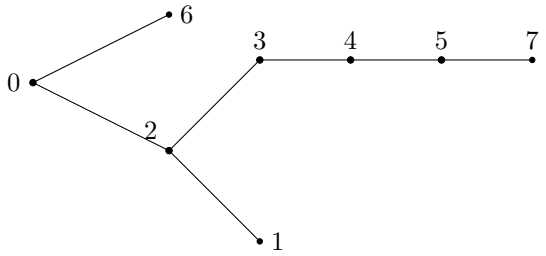
Step 2: Make a copy of the auxiliary tree. Relabel the nonzero vertices of the new tree as follows in preorder.

If the label of the current vertex was k in the auxiliary tree, swap its current label with the label that is currently $(p'_k - a_k)$-th smallest in the subtree rooted at the current vertex. The final tree is $\phi(\mathbf{a})$.

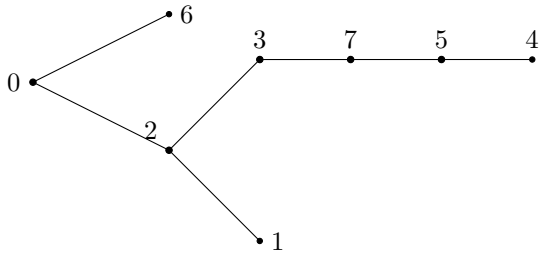
We'll illustrate a few steps. Note that nothing will happen to vertices labeled $0, 6$, and 1. So we start at the vertex 7. Look at $p'_7 - a_6$ which is 2. So we look for the 2nd smallest in the set $\{7, 1, 5, 4, 3, 2\}$, which is 2. So 7 and 2 get swapped. We have the following tree.



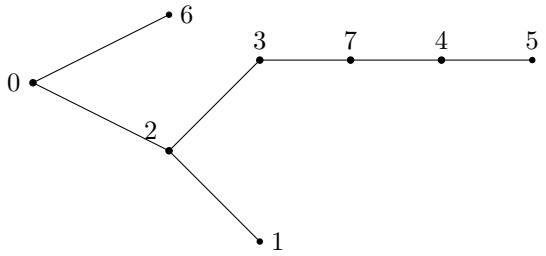
Next we look at the vertex 5. We have $p'_5 - a_5 = 4 - 3 = 1$. So we look for the 1st smallest in the set $\{5, 4, 3, 7\}$, which is 3. So 5 and 3 get swapped. And we have the following tree.



Next we look at the vertex 4. We have $p'_4 - a_4 = 3 - 0 = 3$. So we want the 3rd smallest in the set $\{4, 5, 7\}$, which is 7. So 4 and 7 get swapped. We have the following tree.



Finally we look at the vertex 5. We have $p'_5 - a_5 = 4 - 3 = 1$. So we need the 1st smallest in the set $\{5, 4\}$, which is 4. And we swap 4 and 5 to obtain the tree we wanted.



Note that $D(\mathbf{a}) = 3$. And the final tree we obtained has exactly three inversions, namely $(2, 1), (7, 4)$, and $(7, 5)$. So we have $D(\mathbf{a}) = inv(\phi(\mathbf{a}))$.

5 References

1. Handbook of Enumerative Combinatorics, Miklós Bóna.
2. Parking Functions : From Combinatorics to Probability, Richard Kenyon, Mei Yin.