Addis Ababa University
College Of Natural and Computational Sciences
Department Of Mathematics

Parking Functions and Complete Bipartite Graphs

Medina Suleiman

Advisor: Samuel Asefa Fufa (Ph.D)

A Thesis Submitted in Partial Fulfillment of the Requirement for the Degree of Master of Science in Mathematics

November 26, 2020

Addis Ababa, Ethiopia

Addis Ababa University
Department of Mathematics

This is to certify that the thesis prepared by **Medina Suleiman**, entitled: **Parking Functions and Complete Bipartite Graphs** submit-ted in partial fulfillment of the requirements for the Degree of Master of Science(in Mathematics) compiles with the regulations of the University and meets the accepted standards with respect to originality and quality.

Date:_____

Signed by the Examining Committee:

Supervisor:

1.Dr.Samuel Asefa Fufa _____

(Department of Mathematics, Addis Ababa University, Ethiopia)

Examiners:

1. Dr. Yibeltal Yitayew _____
2. Dr. Yirgalem Tsegaye _____

(Department of Mathematics, Addis Ababa University, Ethiopia)

Chairperson:

1. Dr. Hunduma Legesse _____

(Department of Mathematics, Addis Ababa University, Ethiopia)

Abstract

A Parking function is a sequence of positive integers $(a_1, a_2, \ldots, a_n)$ such that its non-decreasing rearrangement $(b_1, b_2, \ldots, b_n)$, satisfies $b_i \leq i$, for all index i.

In this thesis we will see about parking functions and their interrelation with other combinatorial objects; namely non-crossing partitions and labelled trees. We also discuss on the generalization of parking functions namely; G- parking functions and establish a new relationship between parking functions and spanning trees of complete bipartite graphs.

Contents

List of Figures

Chapter 1

Introduction

Parking functions were first introduced by Konheim and Weiss (1966)[11] as a colorful way to describe their work on computer storage. The parking problem can be described as follows. Suppose that [n] cars labelled $C_1, C_2, \ldots, C_n$ wants to park on a one way street with ordered parking space $(0, 1, 2, \ldots, n-1)$. Further suppose that each car has a preferred parking space i.e, car C_i prefers to park in spot a_i. The cars enter the street one at a time in the order $C_1, C_2, \ldots, C_n$. A car tries to park in its preferred spot if it is unoccupied. If it is occupied then it will drive to the next unoccupied spot. If there is no such space the car leaves the street. The sequence $(a_1, \ldots, a_n)$ is called a parking function of length n if all the cars can park.

Konheim and Weiss (1966)[11] turns out that the number of parking functions of length n is $(n+1)^{n-1}$, this recognizes as the number of tree on a fixed vertex set of cardinality $[n+1]$ or, equivalently the number of spanning tree of the complete graph on $[n+1]$ vertices. There exists simple bijection and other basic techniques showing that the number of tree on $[n+1]$ vertices and the number of parking function of length n are the same.

Cayley's formula immediately noticed that $(n+1)^{n-1}$ is the number of labelled trees on $[n+1]$ vertices. Many bijections between the set of labeled tree and the set of parking functions are constructed. The deep connections between parking functions and other combinatorial structures leads to various applications and generalizations in different fields, like in algebra, probability, statistics, geometry, interpolation theory and representation theory.

In this thesis we will see that a new bijection between Parking functions and complete bipartite graphs and also we will see that the association of parking functions with graphs specially with labelled trees and some generalizations of parking functions. In chapter 2, we review some preliminaries, including graph and graph invariants. In chapter 4, we will present commonly used definitions of parking functions and some examples. In addition, we describe the association of parking functions and graphs .

Chapter 2

Fundamental Concepts

2.1 Graph

Definition 2.1. *A graph G is an ordered triple (V(G),E(G),ψ(G)) consisting of a non-empty set V(G) of vertices a set E(G), disjoint from V(G) of edges and an incident function ψ_E that associates with edge of G an unordered pair of (not necessarily distinct) vertices of G.*

We draw a graph on paper by placing each vertex at a point and representing each edge by a curve joining the locations of its end points.
If e is an edge and 'u' and 'v' are vertices such that $\psi_G(e) = uv$, then e is said to join u and v the vertices u and v are called the end of e.

Example 1. *$G = (V(G), E(G), \psi(G))$, where $V(G) = \{V_1, V_2, V_3, V_4\}$, $E(G) = \{e_1, e_2, \ldots, e_5\}$ and ψ_G is defined by*

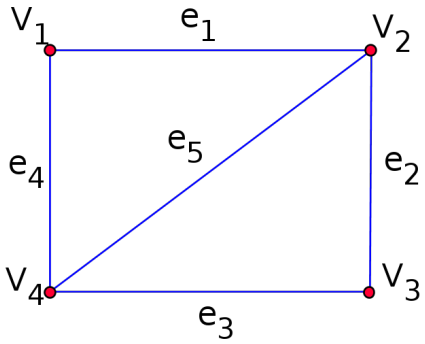


Figure 2.1: Graph of G

$\psi_G(e_1) = V_1V_2$, $\psi_G(e_2) = V_2V_3$
$\psi_G(e_3) = V_3V_4$, $\psi_G(e_4) = V_4V_1$ *and* $\psi_G(e_5) = V_2V_4$

Definition 2.2. *A loop is an edge whose endpoints are equal. Multiple edges are edges having the same pair of endpoints.*

Definition 2.3. *A simple graph is a graph having no loops and multiple edges. We specify a simple graph by its vertex set and edges set, treating the edge set as a set of unordered pairs of*

vertices and writing $e = uv (e = vu)$ for an edge e with endpoints u and v.
When u and v are the endpoints of an edge, they are adjacent and are neighbors. We write $u \longleftrightarrow v$ for "u is adjacent to v".

Definition 2.4. *A subgraph of a graph G is a graph H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$ and the assignment of endpoints to edges in H is the same as in G. We then write $H \subseteq G$ and say that "G contains H".*
A graph G is connected if each pair of vertices in G belongs to a path; otherwise, G is disconnected.

Definition 2.5. *A walk is a list $v_0, e_1, v_1, \ldots e_k, v_k$ of vertices and edges such that, for $1 \leqslant i \leqslant k$, the edge e_i has end points v_{i-1} and v_i. A trail is a walk with no repeated edge. A u,v -walk or u,v -trial has first vertex u and last vertex v; these are its end points. A path is a trail or walk which does not repeat a vertex. The length of a walk, trail or path is its number of edges. A walk is closed if its endpoints are the same.*

Definition 2.6. *A cycle is a closed trail in which the first vertex and last vertex are the same.*

Example 2. *The following graph shows cycles with three and four vertices.*

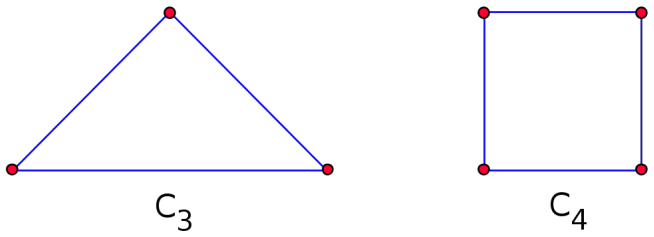


Figure 2.2: Graphs of cycles C_3 and C_4

c_3: cycle with three vertices;

c_4: cycle with four vertices

A graph is acyclic if it does not contain a cycle.

Definition 2.7. *The components of a graph are its maximal connected sub-graph.*

Definition 2.8. *A cut-edge (cut-vertex) is an edge (vertex), which when removed increases the number of components. We write G-e or G-m for the subgroup h of G obtained by deleting an edge e or set of edges m. We write G-v or G-s for the subgraph obtained by deleting a vertex v or set of vertices s.*

Theorem 1. *An edge is a cut-edge if and only if it belongs to no cycle.*

proof 1. *Let e be an edge in a graph G (with endpoints x,y) and let H be the component containing e. Since deletion of e affects no other components, it suffices to prove that H-e is connected if and only if e belongs to a cycle. First suppose that H-e is connected. This implies that H-e contains an x, y-path and this path completes a cycle with e. Now suppose that e lies in*

a cycle c, choose $u, v \in V(H)$, since H is connected, H has a u,v-path p. If p does not contain e, then p exist in H-e. If p contains e suppose by symmetry that x is between u and y on p. Since H-e contains a u, x-path along p, an x,y -path along c and a y,v -path along p, the transitivity of the connection relation implies that H-e has a u,v-path. We did this for all $u, v \in$ V(H), so H-e is connected.

2.1.1 Complete Graph

Definition 2.9. *A complete graph is a graph where every pair of distinct vertices is connected by a unique edge.*

The complete graph on [n] vertices is denoted by K_n, and K_n has $\frac{n(n-1)}{2}$ edges, for every $n \geq 1$. Each vertex of a complete graph has degree n-1.

Example 3. *Examples of complete graph*

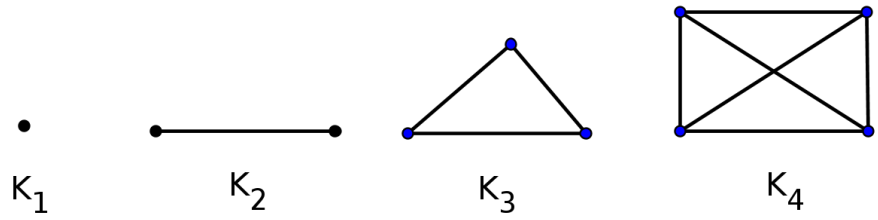


Figure 2.3: Complete Graphs

Lemma 1. *The complete graph on n vertices has $\frac{n(n-1)}{2}$ edges, for $n \geq 1$.*

proof 2. *Suppose that there are [n] teachers and you have to form a committee such that each committee contains 2 teachers. Here assume teachers as vertices and committees as edges. So total number of ways of forming such committees if we have [n] teachers are:*
Use combination to form such committee.

$$\binom{n}{2} = \frac{n!}{(n-2)!2!} \tag{2.1}$$

$$= \frac{n(n-1)(n-2)!}{(n-2)!2!} \tag{2.2}$$

$$= \frac{n(n-1)}{2} \tag{2.3}$$

Therefore K_n has $\frac{n(n-1)}{2}$ edges, for $n \geq 1$.

2.1.2 Tree

Definition 2.10. *A tree is a connected acyclic graph. The edge of a tree are called branches. A leaf (or pendant vertex) is a vertex of degree 1.*

Example 4. *A star is a tree consisting of one vertex adjacent to all the others.*

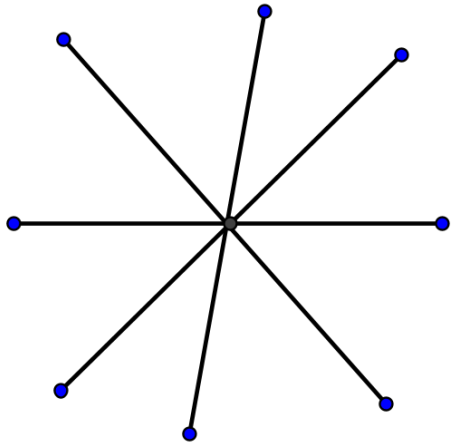


Figure 2.4: Example of a tree

Lemma 2. *Every tree with at least two vertices has at least two leaves. Deleting a leaf from an [n]-vertex tree produces a tree with [n-1] vertices.*

proof 3. *A connected graph with at least two vertices has an edge. In an acyclic graph, an endpoint of a maximal nontrivial path has no neighbor other than its neighbor on the path. Hence the endpoints of such path are leaves. Let v be a leaf of a tree G, and let $G' = G - v$. A vertex of degree 1 belongs to no path connecting two other vertices. Therefore, for $u, w \in v(G')$, every u, w-path in G is also in G'. Hence G' is connected. Since deleting a vertex cannot create a cycle, G' also is acyclic. Thus G' is a tree with $[n-1]$ vertices.*

Theorem 2. *For an n-vertex graph G(with $n \geq 1$,the following are equivalent and characterize the tree with [n] vertices).*
A, G is connected and has no cycles.
B, G is connected and has $[n-1]$ edges.
C, G has $[n-1]$ edges and no cycles.
D, For $u, v \in V(G)$, G has exactly one u, v-path.

proof 4. *We first demonstrate the equivalent of A, B and C by proving that any two of connected, acyclic,$n-1$ edges together imply the third.*
$A \Rightarrow B, C$. We use induction on n. For $n = 1$, an acyclic 1-vertex graph has no edges. For $n > 1$, we suppose that the implication holds for graphs with fewer than n vertices. Given an acyclic connected graph G, Lemma 2 provides a leaf v and states that $G' = G - v$ also is acyclic and connected. Applying the induction hypothesis to G' yields e(G)=n-2 since only one edged is incident to v, we have e(G')=n-1.

$B \Rightarrow A, C$. Delete edges from cycles of G one by until the resulting graph G' is acyclic. Since no edge of a cycle is a cut-edge (Theorem 1) G' is connected. Now the preceding paragraph implies that $e(G') = n-1$. Since we are given $e(G) = n-1$, no edges were deleted. Thus $G' = G$ and G is acyclic.

$C \Rightarrow A, B$. Let $G_1, G_2, G_3, \ldots G_k$ be the component of G since every appears in component, $\sum_i n(G_i) = n$, since G has no cycle, each component satisfies property A. Thus $e(G_i) = n(G_i) - 1$. Summing over i yields $e(G) = \sum_i [n(G_i) - 1] = n - k$. We are given $e(G) = n - 1$, so $k = 1$, and G is connected.

$A \Rightarrow D$. Since G is connected, each pair of vertices is connected by path. If some pair is connected by more than one, we choose a shortest (total length) pair P, Q of distinct paths with the same endpoints. By this extremal choice, no internal vertex of P or Q can belong to the other path. This implies that $P \cup Q$ is a cycle, which contradicts the hypothesis A.

$D \Rightarrow A$. If there is a u, v-path for every $u, v \in V(G)$, then G is connected. If G has a cycle C, then G has two u, v-path for $u, v \in V(G)$; hence G is acyclic.

Corollary 1. *(a) Every edge of a tree is a cut-edge.*
(b) Adding one edge to a tree forms exactly one cycle.
(c) Every connected graph contains a spanning tree(a subtree which contains all vertex of the graph).

proof 5. *(a) A tree has no cycle, so theorem 1 implies that every edge is a cut-edge.*
(b)A tree has a unique path linking each pair of vertices (Theorem 2D), so joining two vertices by an edge creates exactly one cycle.
(c)As in the proof of $B \Rightarrow A, C$ in theorem 2 iteratively deleting edges from cycles in a connected graph yields a connected acyclic subgraph.

Proposition 1. *Every finite tree has at least two vertices of degree 1.*

proof 6. *Notice that any tree must have at least one vertex with degree 1 because if every vertex had degree of at least 2, then one would always be able to continue any walk until a cycle is formed. Now let us assume that we have a tree with exactly one vertex of degree 1. If one removes this vertex of degree 1, the resulting graph must also be a tree since a cycle cannot be added by removing a vertex. In the resulting tree, either the first vertex's neighbor is now of degree 1 or the resulting graph is not a tree and we have a contradiction. If the latter, our proof is done. If the former, we must be able to remove vertices until a contradiction occurs, and there is no vertex of degree 1. We must be able to do this because if we could continue to remove vertices in this way, our graph must be some thing equivalent to a straight path, and a straight path has two vertices of degree 1, namely the endpoints. We have shown that a tree must have at least one vertex of degree 1 and that it cannot have exactly one, so it must have at least two.*

2.1.3 Labeled Tree

Definition 2.11. *A labeled tree is a tree the vertices of which are assigned unique numbers from 1 to n. We can count such trees for small values of n by hand.*

Let T_n be denote the number of labeled trees on n vertices. We know the following values for small n;
$T_2 = 1$, as there is also one tree on 2 vertices
$T_3 = 3$, $T_4 = 16$

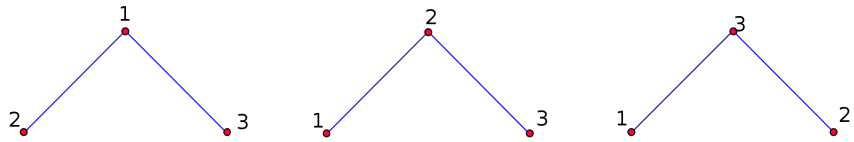


Figure 2.5: Labeled trees on 3 vertices

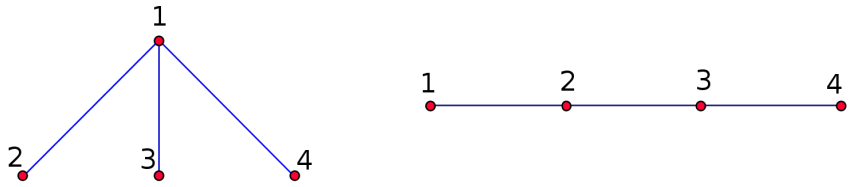


Figure 2.6: Two labeled trees on 4 vertices

Example 5. *The three labeled trees on* 3 *vertices are the following:*

If we count in this fashion we will obtain the following sequence

$$1, 3, 16, 125, 1296, \ldots$$

Definition 2.12. *Prüfer code is a way of encoding of a labeled tree into a sequence of numbers in a unique way.*

For each tree there exists a prüfer code corresponding to it. And for each prüfer code we can restore the original tree.

How to get prüfer code of a tree ?

a, Initialize prüfer code as empty.

b, Start with a leaf of lowest label say x. Find the vertex connecting it to the rest of tree say y, then remove x from the tree and add y to the prüfer code.

c, Repeat the above step 2 until we are left with one edge.

Example 6. *A tree with labels from 1 to 5.*

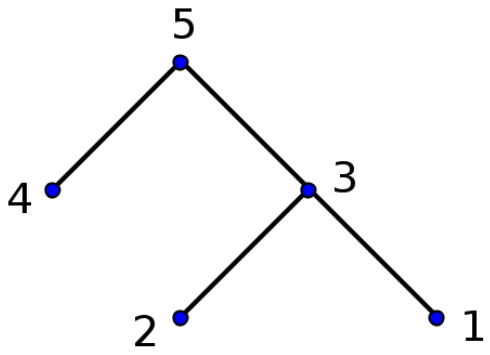


Figure 2.7: A tree with label from 1-5

Prüfer code=$\{\}$
The lowest label leaf is 1, we remove it and add the other vertex (connecting it to the tree) to Prüfer code. the Prüfer code becomes=$\{3\}$
The lowest label leaf is 2, we remove it from tree and add the other vertex (connecting it to the tree) to Prüfer code. The Prüfer code becomes=$\{3, 3\}$
The lowest label leaf is 3, we remove it from tree and add the other vertex (connecting it to the tree) to Prüfer code. The Prüfer code becomes=$\{3, 3, 5\}$
We have only two nodes left now, so we stop.

Therefore $\{3, 3, 5\}$ *is Prüfer code for the given labeled tree.*

Example 7. *The sequence (2,3,4,4) is Prüfer code. Now let us sketch the corresponding tree.*

The given Prüfer code has 4 entries, therefore the corresponding tree will have $4 + 2 = 6$ *vertices.*
The first number in the Prüfer code is 2 and the lowest number not included in the Prüfer code is 1, so we connect 2 and 1. We then drop the leading 2 in the code and put 1 at the back of the code (3,4,4,1). The first number in the code is 3 and the lowest number not included in the Prüfer code is 2, so we connect 2 and 3. Then we drop the leading 3 in the code and put 2 at the back of the code (4,4,1,2)
Now the first number in the code is 4 and the lowest number not included in the Prüfer code is 3, so we connect 4 and 3. Then we drop the leading 4 in the code and put 3 at the back of the code (4,1,2,3). Similarly the first number in the code is 4 and the lowest number not included in the Prüfer code is 5, so we connect 4 and 5. Then the Prüfer code is (1,2,3,5). We have iterated all the way through the code and the two numbers missing are 4 and 6. So we connect them, so the corresponding labeled tree for the given Prüfer code is

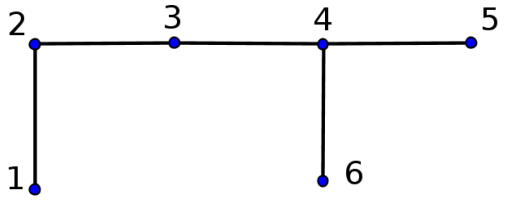


Figure 2.8: The corresponding labeled tree for the given code.

2.1.4 Spanning Tree

Definition 2.13. *A spanning tree for a connected graph G is a tree containing all the vertex of G.*

Example 8. *The original graph and the possible spanning trees that can be created from the original graphs are:*

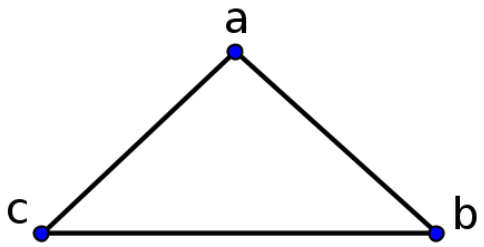


Figure 2.9: Connected graph

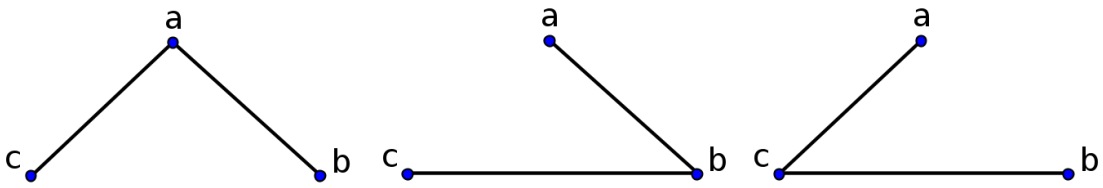


Figure 2.10: All possible spanning trees

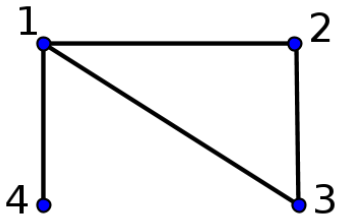
The following are few properties of spanning trees.

1. A connected graph G can have more than one spanning trees.
2. All possible spanning tree of graph G, have the same number of edges and vertices.
3. The spanning tree does not have an cycle(loop).
4. Removing one edge from the spanning tree will make the graph disconnected, i.e. the spanning tree is minimally connected.
5. Adding one edge to the spanning tree will created a circuit or loop, i.e. the spanning tree is maximally acyclic.

Definition 2.14. *In a graph G, contraction of edge e with endpoints u, v is the replacement of u and v with a single vertex whose incident edges are the edges other than e that were incident to u or v. The resulting graph G.e has one less edge than G.*

We can count spanning tree of a graph by a recursive formula. Let $\tau(G)$ be the number of spanning trees of a graph. Then $\tau(G) = \tau(G - e) + \tau(G.e)$, where G.e is an operation known as contracting of an edge.
Cayley derived the well-known formula n^{n-2} for the number of spanning trees in the complete graph K_n.
There are multiple ways to express graphs as trees. One of the most common methods is using the adjacency matrix.

Definition 2.15. *Let G be an undirected graph on $[n]$ labelled vertices, and define an $n \times n$ matrix A by setting $A_{i,j}$ equal to the number of edges between vertices i and j. The matrix A is called the adjacency matrix of G.*



$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Figure 2.11: Undirected graph

Another matrix to represent graph is the incidence matrix.

Definition 2.16. *Let G be a directed graph with out loops. Let $\{V_1, \ldots, V_n\}$ be denote the vertices of G, and let $\{e_1, \ldots, e_m\}$ denote the edges of G. The incidence matrix of G is the $n \times m$ matrix A defined by*

$a_{i,j} = 1$ *, If V_i is the head of e_j*
$a_{i,j} = -1$ *, If V_i is the tail of e_j*
$a_{i,j} = 0$ *, Otherwise*

Example 9.

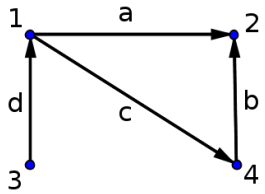


Figure 2.12: Directed graph

$$\begin{bmatrix} -1 & 0 & -1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & -1 & 1 & 0 \end{bmatrix}$$

With the ordering $1, 2, 3, 4/a, b, c, d$.

Theorem 3. *Let G be a directed graph with out loops, and let A be the incidence matrix of G. Remove any row from A, and let A_0 be the remaining matrix. The number of spanning trees of G is determinant of $A_0 A_0^T$.*

Definition 2.17. *If G is a graph on n vertices with $V = \{V_1, \ldots, V_n\}$ then its graph Laplacian L is an $n \times n$ matrix whose entries are*
$L_{i,j} = deg(V_i)$ *, If i=j*
$L_{i,j} = -1$ *, If i is different from j and if V_i and V_j are adjacent in G*
$L_{i,j} = 0$ *, Otherwise*

Theorem 4. *(Kirchoff's Matrix-Tree Theorem). If G is an undirected graph and L is its graph Laplacian, then the number N_T of spanning trees contained in G is given by the following computation.*

1, Choose a vertex V_j and eliminate the j^{th} row and column from L to get a new matrix L_j
2, Compute $N_T = det(L_j)$

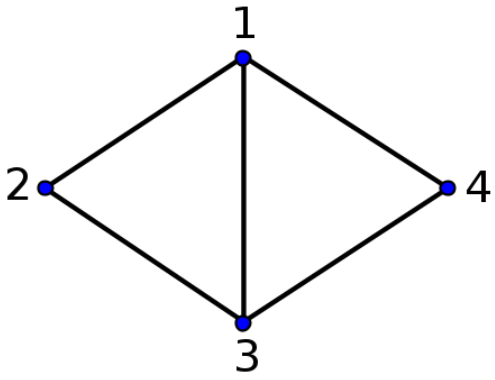


Figure 2.13: Undirected graph

Example 10. *Find number of spanning trees of the above graph G(figure 2.13).*

1, First we construct the Laplacian matrix Q for the graph G, so

$$Q = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}$$

Next we construct matrix Q' by deleting any row r and column c (where $r = c$ or $r \neq c$) from Q. Let us delete row 1 and column 1.

$$Q' = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Finally, we take the determinant of Q' to obtain the number of spanning trees of the graph. Here in the given graph number of distinct spanning trees are 8.

2.1.5 Bipartite Graph

A bipartite graph is a special kind of graph.

Definition 2.18. *A graph G is bipartite if the vertex set of G can be partition into two sets A and B such that if the vertex u and v are in the same set,u and v are non-adjacent.*

Proposition 2. *Every tree is a bipartite graph.*

proof 7. *We may use the structural theorem on trees to prove this fairly easily by induction. Clearly when $|T| = 1$,T is bipartite (although one of the parts will be empty). For the inductive step,let $|T| = n$,and let V be a leaf; T-V is a tree by the structural theorem,and by the inductive hypothesis,T-V is bipartite. Thus,all vertices of T except for V may easily be classified into partitions satisfying the non-adjacency criterion. Since $d_T(V) = 1, V$ has one neighbor. Put V in the partition not occupied by its neighbor to satisfy the non-adjacency condition.*

Lemma 3. *Every closed odd walk contains no odd cycle.*

proof 8. *We use induction on the length l of a closed odd walk W. Basic step: $l = 1$. A closed walk of length 1 traverses a cycle of length 1.*

Induction step; $l = 1$. Assume the claim for closed odd walk shorter than W. If W has no repeated vertex(other than first=last), then W itself forms a cycle of odd length. If vertex V is repeated in W, then we view W as starting at V and break W into two U, V-walk. Since W has odd length, one is shorter than W. By the induction hypothesis, it contains an odd cycle, and cycle, and this cycle appears in order in W.

Theorem 5. *A graph is bipartite if and only if it has no odd cycle.*

proof 9. *Necessity: Let G be a bipartite graph. Every walk alternates between the two sets of a bipartition, so every return to the original partite set happens after an even number of steps. Hence G has no cycle.*

Sufficiency: Let G be a graph with no odd cycle. We prove that G is bipartite by constructing a bipartition of each nontrivial component. Let u be a vertex in a non trivial component H. For each $V \in V(H)$, let f(V) be the minimum length of a U, V-path. Since H is connected, f(V) is defined for each $V \in V(H)$ Let $X = \{V \in V(H) : f(V) iseven\}$ and $Y = \{V \in V(H) :$ f(V) is odd $\}$.an edge U,V with in x or y would create a closed odd walk using a shortest U, V-path, the edge VV' and the reverse of a shortest U,V-path,path by lemma 3 such a walk must contain an odd cycle, which contradicts our hypothesis. Hence X and Y are independent sets. Also $X \cup Y = V(H)$, so H is an X, Y-bigraph.

Few important properties of bipartite graph.

1. Bipartite graphs are 2 colorable.
2. Bipartite graphs contains no odd cycles.
3. Every subgraph of a bipartite graph is itself bipartite.
4. In any bipartite graph with bipartition X and Y sum of degree of vertices of set X is equal to sum of degree of vertices of a set Y.

Definition 2.19. *A complete bipartite graph is a special kind of bipartite graph where every vertex of the first set is connected to every vertex of the second set.*

A complete bipartite graph with $|A| = m$ and $|B| = n$, is denote $K_{m,n}$.

2.2 Cayley's Formula

Cayley's formula tells us how many different trees we can construct on [n] vertices. We can think about this process as beginning with n vertices and then placing edges to make a tree. Another way to think about it involves beginning with the complete graph on n vertices K_n,and the removing edges in order to make a tree, cayley's formula tells us how many different ways we can do this. There are called spanning trees on [n] vertices and we will denote the set of these spanning trees by T_n.

Theorem 6. *(Cayley's) There are n^{n-2} labelled trees with n vertices,$n \geq 2$.*

proof 10. *Let T be a tree with n vertices and let the vertices be labelled $1, 2, \ldots n$. Remove the pendant vertex (and the edge incident to it) having the smallest label ,say U_1. Let V_1 be the vertex adjacent to U_1. From the remaining $[n-1]$ vertices. Let U_2 be the vertices the pendant vertex with the smallest label and let V_2 be the vertex adjacent to U_2 . We repeat this operation on the remaining $n-2$ vertices,then on $n-3$ vertices, and so on. This process completes after $n-2$ steps,when only two vertices are left.*

Let the vertices after each removal have labels $V_1, \ldots, V_{n-2}$. Clearly, the tree T uniquely defines the sequence.

$(V_1, V_2, V_3, \ldots V_{n-2})$ ..... *(1)*

Conversely,given a sequence of $n-2$ labels,an n-vertex tree is constructed uniquely as follows. Determine the first number in the sequence

$\{1, 2, 3, \ldots, n\}$ ..... *(2)*

that does not appear in the (1). Let this number be U_1. Thus the edge (U_1, V_1) is define. Remove V_1 from the (1) and U_1 from the (2). In the remaining sequence of the(2), find the first number which does not appear in the remaining sequence of the (2). Let this be U_2 and thus the edge (U_2, V_2) is defined. The construction is continued till the first sequence has no element left. Finally, the last two vertices remaining in the (2) are joined. For each of the $n-2$ elements in the (1), we choose any one of the n number, thus forming $n^{n-2}(n-2)$-tuples, each defining a distinct labelled tree of n vertices. Since each tree defines one of these sequence uniquely, there is a one-one correspondence between the tree and the n^{n-2} sequence.

Chapter 3

Literature Review

Parking functions were introduced by Konheim and Weiss (1966) [11] to study the widely used storage device of hashing. They showed there are $(n+1)^{n-1}$ parking functions, for example when $n = 3$, the number of parking function is 16.
Since then parking functions have appeared all over combinatorics; in the hyperplane arrangements (Shi 1986), in the analysis of set partitions (Stanley, 1997 b), Chip firing games (Cori and Rossin, 2000), in the enumerative theory of tree and forests (Chassaing and Marcker, 2001) poly topes (Stanley and Pitman, 2002)(Chebikin and Postnikov, 2010), Yan (2015) gives an accessible extensive survey which may be supplemented by Stanley (1999 a).

Theorem 7. *The number of parking sequence of $f(\overrightarrow{y})$ for car sizes $\overrightarrow{y} = (y_1, \ldots, y_n)$ is given by the product*

$$f(\overrightarrow{y}) = (y_1 + n) \cdot (y_1 + y_2 + n - 1) \cdots (y_1 + \cdots + y_{n-1} + 2).$$

Consider $M = y_1 + y_2 + \ldots + y_n + 1$ parking spaces arranged in a circle. We will consider parking cars on this circular arrangements, with out a cliff for cars to fall off. Observe that when all the cars have parked, there will be one empty spot left over. We claim that there are

$$Mf(\overrightarrow{y}) = (y_1 + n)(y_1 + y_2 + n - 1) \cdots (y_1 + \cdots + y_n + 1) \tag{3.1}$$

such circular parking sequences. The first car C_1 has M ways to choose its parking spot.

The next step is counter intuitive. After car C_1 has parked, erase the marking for the remaining $y_2 + \cdots + y_n + 1$ spots and put in $n + 1$ dividers. These dividers create $n + 1$ intervals on the circle, where one interval is taken up by C_1. Furthermore, these dividers are on wheels and can freely move along the circle. Each interval will accept one car. For example, consider the case where $n = 5$ and $\overrightarrow{y} = (2, 5, 1, 3, 2)$ so that $M = 2+5+1+2+3+1 = 14$, and $c_1 = 5$.

We will now create a circular parking sequence, but only at the end do we obtain the exact positions of cars C_2 through C_{n+1}. That is, instead of focusing on the number of specific spot preference each car could have, we keep track of the order the cars park in, which will then determine the exact locations of the cars.

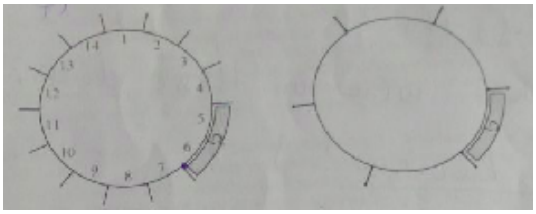


Figure 3.1: The circular parking sequence.

The second car has two options. The first is that it has a desired position already taken by C_1. In this case, it will cruise until the next empty spot. This can happen in y_1 ways, and then car C_2 obtains the next open interval after the interval C_1 is in. Otherwise, the car C_2 has a preferred spot not already taken. In this case C_2 has n open intervals to choose from. The total number of options for C_2 is $y_1 + n$.

The third car C_3 has the same options. First, it may desire a spot that is already taken, in which case it will have to cruise until the next open interval. This can happen in $y_1 + y_2$ ways. Note that this count applies to both the case when C_1 and C_2 are parked next to each other, and when C_1 and C_2 have open intervals between them. Otherwise, C_3 has $n-1$ open intervals to pick from.
In general, car C_i has $y_1 + \cdots + y_{i-1} + n + 2 - i$ choices. This pattern continues up to C_n, which has $y_1 + \cdots + y_{n-1} + 2$ possibilities. For example, suppose C_2 and C_3 in our above example have parked as below.

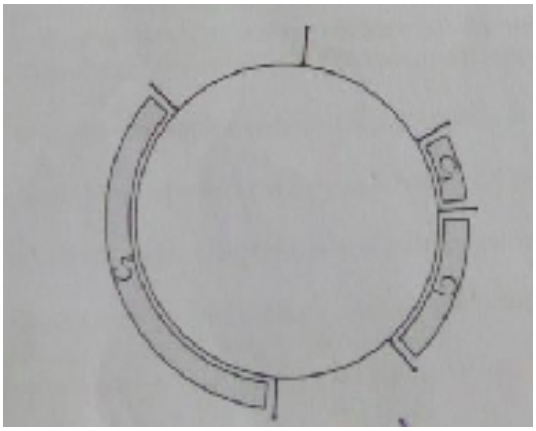


Figure 3.2: The circular parking sequence

Then C_4 may either cruise on C_1 and C_3 (in $y_1 + y_3$ ways), it may cruise on C_2(in y_2 ways), or it can pick one of the three available intervals directly. In total, C_4 has $(y_1 + y_3) + y_2 + 3 = 11$ ways to park.
One can imagine that when we park a car, We do not set the parking brake, but put the car in neutral, so that the car and the dividers can move as necessary to make room for future cars. Thus the total number of circular parking arrangements of this type is $M \cdot (y_1 + n) \cdot (y_1 + y_2 + n - 1) \cdots (y_1 + \cdots + y_{n-1} + 2)$, where the i^{th} factor is the number of options for the car C_i. This proves the claim about the number of circular parking sequence in (3.1).

Hence, to prove theorem 7 we need only observe that the circular parking sequences with spot M empty are the same as our parking sequence. This follows from the observation that no car in the circular arrangement has preference M, since otherwise this spot would not be empty. Furthermore, no car would cruise by this empty spot.

Observe that the set of circular parking sequence is invariant under rotation. This is, if $(C_1, C_2, \ldots, C_n)$ is parking sequence, then so is the sequence $(C_1 + a, C_2 + a, \ldots C_n + a)$, where all the additions are modulo M. In particular, the number of circular parking sequence with spot M empty is given by $1/M \cdot Mf(\overrightarrow{y}) = f(\overrightarrow{y})$.

The idea of considering a circular arrangement goes back to Pollak;see [17]. In fact, when all the cars have size 1, this argument reduced to his argument that the number of classical parking functions is $(n+1)^{n-1}$. A simple and elegant proof was given by Pollak.

Theorem 8. *(Pyke, 1959; Konheim and Weiss, 1966)*
Let f(n) be the number of parking functions of length n. Then $f(n) = (n+1)^{n-1}$.

Proof. (Pollak, C.1974). Add an additional space $n+1$ and arrange the space in a circle. Allow $n+1$ also as a preferred spaces.
Now all cars can park, and there will be one empty space. α is a parking function $\Longleftrightarrow$ if the empty space is $n+1$. If $\alpha = (a_1, a_2, \ldots, a_n)$ leads to car C_i parking at space P_i, then $(a_1 + j, \ldots, a_n + j)$(modulo $n+1$) will lead to car C_i parking at space $P_i + j$. Hence exactly one of the vectors $(a_1 + i, a_2 + i, \ldots, a_n + i)$(modulo $n+1$) is a parking function, so $f(n) = \frac{(n+1)^n}{n+1} = (n+1)^{n-1}$. □

The number $(n+1)^{n-1}$ also appears in the study of two other combinatorial structures, and they are non-crossing partitions and hyperplane arrangements. It is known that the set of non-crossing partitions of $\{1, 2, 3, \ldots, n+1\}$ is a sub poset of the poset of partition of $\{1, 2, 3, \ldots, n+1\}$. It turns out that the number of maximal chains in the poset of non-crossing partitions is $(n+1)^{n-1}$.
A non crossing partition of the set $[n] = \{1, 2, 3, \ldots, n\}$ is a partition Π of the set [n] with the property that if $a < b < c < d$ and some block B of Π contains both a and c, while some block B' of Π contains both b and d, then B=B'. The study of non-crossing partitions goes back at least to H.W. Becker [1] where they are called "polar rhyme scheme". The systematic study of non-crossing partition began with Kreweras in 1972 [12] and Poupard [16], it caught the imagination of combinatorial lists beginning in the 1980s and has come to regarded as one of the standard objects in the field.

The notion of non-crossing partition has a rich and venerable history with in enumerative combinatorics. Whether as the main object of study as an object where a theorem or theory applies, or as a mathematical object serving as a model for a phenomenon of independent interest, non-crossing partition appear to support a rich amount of mathematics in and beyond enumerative combinatorics .
The number of non-crossing partitions of $\{1, 2, 3, \ldots, n\}$ is the catalan number $C_n = \frac{1}{n+1}\binom{2n}{n}$
.

Foata and Riordan found that the parking functions of size n are in bijection with labeled tree with $[n+1]$ nodes, a labeled tree is a tree the vertices of which are assigned unique numbers from 1 to n. Beissinger and Peled linked Krewera's polynomial family to inversions and external activity in labeled trees [2]. It is well known that the set of parking functions of size n and the set of rooted labeled forests with n vertices are in bijection. A rooted(labeled) forest is a labeled graph where the connected components are trees with a distinguished vertex or root. By joining to the graph a new vertex connected with the roots of all trees,we construct a bijection, between the set of rooted forest with n vertices and the set of trees with $[n+1]$ vertices, known to have $(n+1)^{n-1}$ elements by Cayley's theorem.

There is a particular generalization of parking functions that has been considered recently. Returning to the parking process described at the beginning of this thesis, the essential features are as follows:

1. Each car has an initial preferred parking location.
2. If a location is currently occupied then cars have a consistent rule to proceed and search for an opening.
3. Each car will terminate its search after finitely many steps and exit if it has not already found a place to park.

For the path we start at a vertex and then if we cannot find a space we continue moving along the path and eventually exit. If we make the last vertex the root of the path, then the parking rule can be stated as follows: go to the preferred vertex, if it is not available continue moving towards the root and park at the first available spot; if no spot has been found and the root is reached, then exit. This rule works for any tree T. This suggests that one can consider parking functions on a general rooted tree T, where each vertex of T represents a parking space, and edges are oriented towards the root, representing one way streets. Again n cars enter one by one, each with a preferred parking space. A sequence of preferences such that all cars can park in the vertices of T under the above rule is called a T-parking function and was first studied by Burner and Panholzer [14]. Who also considered parking functions on mappings. Via analytic combinatorial techniques, Burner and Panholzer revealed a beautiful relation between the total number of parking functions on rooted trees and that on mappings, presented exact and asymptotic enumerative results, and described a phase change behavior when one considers m cars parking on n available spots.

Let T be a rooted tree with vertices labeled $\{1, 2, 3. \ldots, n\}$. Let $\{C_1, \ldots, C_n\}$ be the n cars assume that C_i prefers the vertex with label U_i. The sequence $\{U_1, U_2, U_3, \ldots, U_n\}$ is a T-parking function if and only if all the cars can find a parking space in T.

pr*ü*fer code is a way of encoding a labeled tree with [n] vertices using a sequence of [n-2] integers in the interval [0,n-1]. This encoding also acts as a bijection between all spanning trees of a complete graph and the numerical sequences. Pr*ü*fer codes are used mostly in solving combinatorial problems. For each tree there exist a pr*ü*fer code corresponding to it and for each pr*ü*fer code we can restore the original tree. It follows that also every pr*ü*fer code (i.e. a sequence of n-2 numbers in the range [0,n-1]) corresponds to a tree. Therefore all trees and

all prüfer codes form a bijection. Prüfer code leads to deeper understanding of the internal structure of the parking functions and their relationship to other combinatorial objects, such as trees. Each parking function can be transformed into a corresponding prüfer code, and each prüfer code corresponds to exactly one labeled trees.

Theorem 9. *Let $S \subseteq N$ with $|S| \geq 2$. There is a bijection between $S^{|S|-2}$ and the set of all trees with vertex set S.*

proof 11. *Let $n = |S|$. Consider the function f produced by the prüfer code algorithm. To show that f is the desired bijection. This will follow if we show that every sequence $(a_1, a_2, a_3, \ldots, a_{n-2}) \in S^{n-2}$ defines a unique tree T such that $f(T) = (a_1, a_2, a_3, \ldots, a_{n-2})$.*

If n=2, then there is exactly one tree on 2 vertices and the algorithm always outputs the empty sequence, the only such sequence. So the claim clearly holds for $n = 2$.
Now assume $n > 2$ and the claim by induction holds for all sets S' of size less than n.

Consider a sequence $(a_1, \ldots, a_{n-2}) \in S^{n-2}$.
We need to show that $(a_1, \ldots, a_{n-2})$ can be uniquely produced by the algorithm. Let's analyze this situation. Suppose that the algorithm produced $f(T) = (a_1, \ldots, a_{n-2})$ for some tree T. Then non of $(a_1, \ldots, a_{n-2})$ is a leaf of T. Indeed, when a vertex is set to be a_i it is adjacent to a leaf with T_i. So if a_i is a leaf of T_i, then T_i has only 2 vertices. However T_i has $|S| - i + 1$ vertices, which is ≥ 3, since $i \leq |s| - 2$. This implies that the label of the first leaf removed from (the hypothetical tree) T is precisely the minimum element of the set $S \backslash (a_1, \ldots, a_{n-2})$. Let v be this element. In other words, in every tree T such that $f(T) = (a_1, \ldots, a_{n-2})$ the vertex v is a leaf whose unique neighbour is a_1. By induction, there is a unique tree T' with vertex set $S \backslash \{v\}$ such that $f(T') = (a_2, \ldots, a_{n-2})$. Adding the vertex v and the edge $\{a_1, v\}$ to T' yields the desired unique tree T with $f(T) = (a_1, \ldots, a_{n-2})$.

The base concept of parking function has also been extended to more generalized objects. Stanley proposed "K-parking function", which consist of a map from [n] to [kn] such that for all $i \leq n$. The number of $a_j \leq (i-1)k+1$ is grater than or equal to i and this concept was further expanded and explored by Yan [21]. (Alternatively if b_i are the a_i arranged in a non-decreasing order, the $b_i \leq (i-1)k+1$).

Gilbey and Kalikow added "type A" with a preferred subset of spaces to create valet functions [9]; valet parking is a parking service offered by some restaurants, stores and other businesses. Biane generalized to parking function of "Type B" where the function only required $a_i \leq n$ and showed that this generalized the relation ship between parking function and non-crossing partition given earlier by Stanley [3]. Yan has done further exploration of various aspects of parking functions and their relation to other structures (Yan and Kostic [20], Kung and Yan [13]).

Chapter 4

Parking Functions and Some Combinatorial Objects

Definition 4.1. *Let $\overrightarrow{a} = (a_1, a_2, a_3, \ldots, a_n)$ be a sequence of positive integers, and let $b_1 \leq b_2 \leq b_3, \ldots, \leq b_n$ be the non-decreasing rearrangement of $\overrightarrow{a}$. Then the sequence $\overrightarrow{a}$ is a parking function if and only if $b_i \leq i$ for all indexes i.*

Car C_i prefers space a_i. If a_i is occupied, then C_i takes the next available space. We call $(a_1, a_2, a_3, \ldots, a_n)$ a parking function of length n if all cars can park. It is well known that the number of such parking functions is $(n+1)^{n-1}$.

Example 11. *For n=2, there are 3 different parking functions (1,1),(1,2),(2,1).*
For n=3, there are sixteen different parking functions $(1,1,1), (1,1,2), (1,2,1), (2,1,1), (1,1,3), (1,3,1), (3,1,1), (1,2,2), (2,1,2), (2,2,1), (1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1)$.

Example 12. *The sequence (3,1,4,3) is not a parking function since the last car will go to space 3, but space 3 and 4 are already occupied.*

Definition 4.2. *Let there be n cars $C_1, C_2, C_3, \ldots, C_n$ of sizes $z_1, z_2, z_3, \ldots, z_n$, where $z_1, \ldots, z_n$ are positive integers. Assume there are $\sum_{j=1}^{n} z_j$ spaces in a row. Furthermore, let car C_j have the preferred spot a_j. Now let the cars in the order C_1 through C_n park according to the following rule.*

Starting at positions a_j, car C_j looks for the first empty spot $i \geq a_j$. If the spaces i through $i + z_j - 1$ are empty, then car C_j parks in these spots. If any of the spots i+1 through $i + z_j - 1$ is already occupied, then there will be a collision, and the result is not a parking sequence.

Iterate this rule for all the cars $C_1, \ldots, C_n$. We call $(a_1, \ldots, a_n)$ a parking sequence for $\overrightarrow{z} = (z_1, \ldots, z_n)$ if all n cars can park without any collisions and with out leaving the $\sum_{j=1}^{n} z_j$ parking spaces.

Example 13. *Let $\overrightarrow{z} = (2,2,1)$ be sizes of three cars with preferences $\overrightarrow{a} = (2,3,1)$. Then there are $2+2+1=5$ available parking spaces, and the final configuration of the cars is as follows.*
All cars are able to park, so this yields a parking sequence.

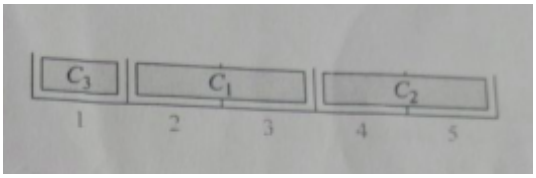


Figure 4.1: Parking sequence

There are two ways in which a sequence can fail to be a parking sequence. Either a collision occurs, or a car passes the end of the parking lot. For example consider three cars with $\overrightarrow{z} = (2, 2, 2)$ *and preferences* $\overrightarrow{a} = (3, 2, 1)$*. Then we have* $2 + 2 + 2 = 6$ *parking spots, and the first car can parks in its desired spot. However, the second car prefers spot 2, and since spot 2*

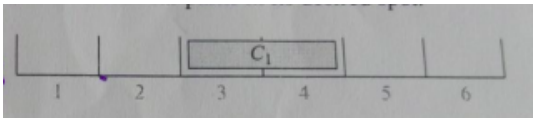


Figure 4.2: Parking sequence

is open, he tries to take spots 2 and 3, but collides with C_1 *in this process. Hence, this is not a parking sequence.*

If we had $\overrightarrow{z} = (2, 2, 2)$ *and* $\overrightarrow{a} = (2, 5, 5)$*, then the first two cars are able to park with no difficulty. But car* C_3 *will pass by all the parking spots after his preferred spot without seeing an empty spot. Hence, this also fails to be a parking sequence.*

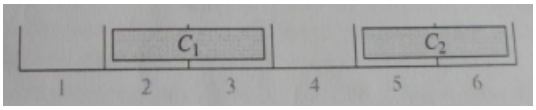


Figure 4.3: Parking sequence

The classical notion of a parking function is obtained when all the cars have size 1; that is $\overrightarrow{z} = (1, 1, \ldots, 1)$. Note in this case there are no possible collisions.

4.1 Non-Crossing Partition and Parking Functions

Richard Stanley, in his article of 1996, showed that there is a bijection between the set of parking functions and the set of maximal chains in the lattice of non-crossing partitions. He gave an algorithm that injectivity assigns to each maximal chain a parking function, and then relied on the fact that the two sets have the same cardinality to deduce surjectively.

Definition 4.3. *A non-crossing partition of* $\{1, \ldots, n\}$ *is a partition* $\{B_1, \ldots, B_k\}$ *of* $\{1, \ldots, n\}$ *such that* $a < b < c < d$*,* $a, c \in B_i$*,* $b, d \in B_j \implies i{=}j$*.*

Example 14. *The first partition is non-crossing partition with* 134/578/2/6*, but the second partition is crossing partition with* 1458/237/6*.*

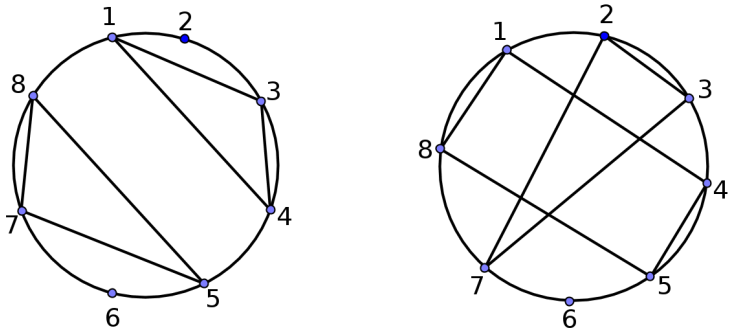


Figure 4.4: Non- crossing and crossing partition

Example 15. *Let* $n = [4]$, $1-2-3-4, 1-2-34, 1-23-4, 12-3-4, 13-2-4, 14-2-3, 1-24-3, 123-4, 124-3, 134-2, 1-234, 1234$ *these are non-crossing partition of [4].*

The number of non-crossing partitions of $\{1, 2, 3, \ldots, n\}$ is the Catalan number $C_n = \frac{1}{n+1}\binom{2n}{n}$.

Definition 4.4. *A maximal chain m of non-crossing partitions of* $[n+1]$ *is a sequence* $\{\Pi_0, \Pi_1, \ldots \Pi_n\}$ *of non-crossing partitions of* $[n+1]$ *such that* Π_i *is obtained from* Π_{i-1} *by merging two blocks into one (Hence* Π_i *has exactly n+1-i blocks).*

Example 16. *Let* $n = [4]$, *then* $[n+1] = [5] = \{1, \ldots, 5\}$. *Then* $1-2-3-4-5, 1-25-3-4, 1-25-34, 125-34, 12345$ *is a maximal chain.*

Define: min B = least element of block B and
$j < B : j < k$, $\forall k \in B$.
Suppose π_i is obtained from π_{i-1} by merging together blocks B and B' with $minB < minB'$.
Then we define
$\lambda_i(m) = max\{j \in B : j < B'\}$
$\lambda(m) = (\lambda_1(m), \ldots, \lambda_n(m))$

Example 17. *Let* $n = [3]$. *Then the following are the maximal chains of* $[n+1] = [4]$.

1. $1-2-3-4, 12-3-4, 123-4, 1234$: $\lambda(m) = (1, 2, 3)$
2. $1-2-3-4, 12-3-4, 12-34, 1234$: $\lambda(m) = (1, 3, 2)$
3. $1-2-3-4, 12-3-4, 124-3, 1234$: $\lambda(m) = (1, 2, 2)$
4. $1-2-3-4, 13-2-4, 134-2, 1234$: $\lambda(m) = (1, 3, 1)$
5. $1-2-3-4, 13-2-4, 123-4, 1234$: $\lambda(m) = (1, 1, 3)$
6. $1-2-3-4, 14-2-3, 134-2, 1234$: $\lambda(m) = (1, 1, 1)$
7. $1-2-3-4, 14-2-3, 124-3, 1234$: $\lambda(m) = (1, 1, 2)$
8. $1-2-3-4, 14-2-3, 14-23, 1234$: $\lambda(m) = (1, 2, 1)$

9. $1-2-3-4, 1-23-4, 14-23, 1234$ *:* $\lambda(m) = (2,1,1)$

10. $1-2-3-4, 1-23-4, 1-234, 1234$ *:* $\lambda(m) = (2,3,1)$

11. $1-2-3-4, 1-23-4, 123-4, 1234$ *:* $\lambda(m) = (2,1,3)$

12. $1-2-3-4, 1-24-3, 124-3, 1234$ *:* $\lambda(m) = (2,1,2)$

13. $1-2-3-4, 1-24-3, 1-234, 1234$ *:* $\lambda(m) = (2,2,1)$

14. $1-2-3-4, 1-2-34, 134-2, 1234$ *:* $\lambda(m) = (3,1,1)$

15. $1-2-3-4, 1-2-34, 1-234, 1234$ *:* $\lambda(m) = (3,2,1)$

16. $1-2-3-4, 1-2-34, 12-34, 1234$ *:* $\lambda(m) = (3,1,2)$

Theorem 10. *λ is a bijection between the maximal chains of non-crossing partitions of $[n+1]$ and parking functions of length n.*

Corollary 2. *(Kreweras, 1972): The number of maximal chains of non-crossing partitions of $[n+1]$ is $(n+1)^{n-1}$.*

Theorem 11. *There is a bijection between the maximal chains of non-crossing partitions of $[n+1]$ and rooted forests on n.*

proof 12. *Since we already have a bijection from maximal chains of non-crossing partitions of $[n+1]$ and parking functions of length n and also we already have a bijection between parking function of length n and rooted forest on n, this gives us a bijection between the maximal chains of non-crossing partitions of [n+1] and rooted forest on n.*

Example 18. *let $m = 1-2-3, 13-2, 123$ be the maximal chain of $n = [3]$. Then it corresponds to the rooted tree.*

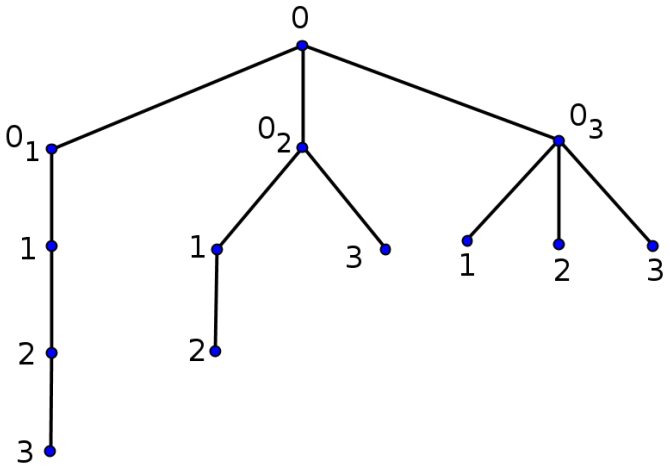


Figure 4.5: Rooted tree

In general there are 3 maximal chains of [3],
(a), $1-2-3, 12-3, 123$ *(b),* $1-2-3, 1-23, 123$ *(c),* $1-2-3, 13-2, 123$

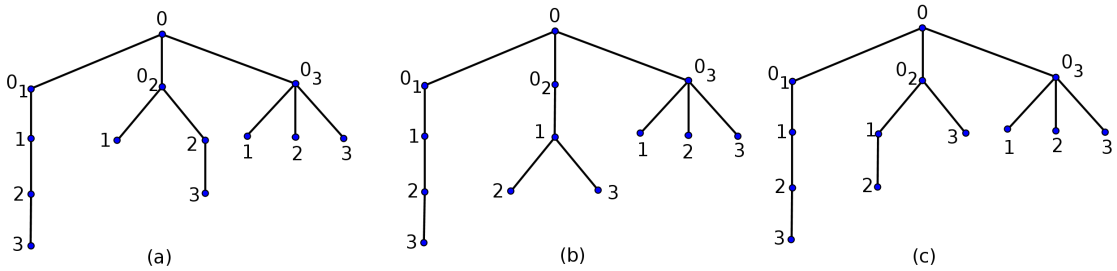


Figure 4.6: The corresponding rooted tree for the given maximal chains

4.2 Parking Functions Associated with Graphs

One interpretation of classical parking function is drivers attempting to park in a preferred spot along a street and parking in the first available spot they find afterwards. We may extend this notion to general directed graphs by allowing drivers to choose which out-edge to travel along. G-parking function are a broad generalization of the classical parking functions.

Example 19. $P = (1, 3, 1, 4, 2)$

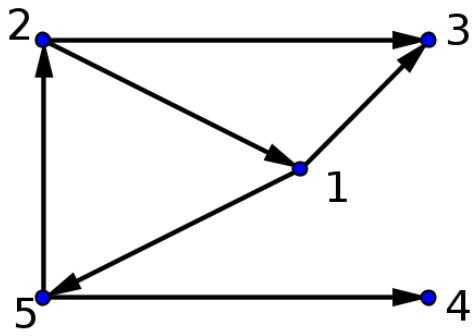


Figure 4.7: Directed graph

4.2.1 G-Parking Functions

Definition 4.5. *Let G be a directed graph on the vertices $\{0, 1, 2, \ldots, n\}$. We allow G to have multiple edges, but not loops. A G-parking function is a sequence $\{a_1, \ldots, a_n\}$ of non-negative integers that satisfies the following conditions: for each subset $L \subseteq \{1, 2, 3, \ldots, n\}$ of vertices of G, there exists a vertex $j \in L$ such that the number of edges from j to vertices outside of L is greater than a_j. For the complete graph $G = K_{n+1}$, these are the classical parking functions.*

Let G be directed graph on vertices $\{0, \ldots, n\}$. We allow G to have multiple edge but not loops. To distinguish between multiple edges of G, we fix an order on the set of edges going from i to j for all $i \neq j$.
A subtree of G rooted at k is a subgraph T of G containing k such that for every vertex i of T, there is a unique path in T from i to k. Spanning tree is a subtree which contains all vertices of G.

Let I_G be the set of subtrees of G rooted at 0, and let S_G be the set of spanning trees of G rooted at 0. All spanning trees in this thesis are assumed to be rooted at 0. Let P_G be the set of G-parking functions.
For every $T \in I_G$, let $\Pi(T)$ be a total order on the vertices of T, and write $i <_{\Pi(T)} j$ to denote that i is smaller than j in this order. We call the set $\Pi(G) = \{\Pi(T) | T \in I_G\}$ a proper set of tree orders if the following conditions hold for all $T \in I_G$:

1. If (j,i) is an edge of T, then $i <_{\Pi(T)} j$;
2. If t is a subtree of T rooted at 0, then the order$\pi(t)$ is consistent with $\Pi(T)$; in other words, $i <_{\Pi(t)} j$ if and only if $i <_{\Pi(T)} j$ for $i, j \in t$.

For $T \in I_G$ and a vertex j of G the order $\Pi(T)$ induces the order on the edges going from j to vertices of T in which (j,i) is smaller than (j,i') whenever $i <_{\Pi(T)} i'$ and which is consistent with the previously fixed order on multiple edges. We write $e <_{\Pi(T)} e'$ to denote that e is smaller than e' in this order.

Given a proper set of tree orders $\Pi(G)$, define map $f_{\Pi,G} : S_G \longrightarrow P_G$ as follows. For $T \in S_G$ and a vertex $j \in \{1, 2, 3, \ldots, n\}$, let e_j be the edge of T going out of j. Set $f_{\Pi,G}(T) = (a_1, \ldots, a_n)$, where a_j is the number of edges e going out of j such that $e <_{\Pi(T)} e'$. For the rest of the section, we write f instead $f_{\Pi,G}$.

Theorem 12. *The map f is bijection between S_G and P_G.*

proof 13. *We begin by checking that $f(T)$ is a G-parking function.*

Lemma 4. *$f(T) \in P_G$ for $T \in S_G$.*

proof 14. *For a subset $L \subseteq \{1, 2, 3, \ldots, n\}$, let j be the smallest vertex of L in the order $\Pi(T)$. Let $e_j = (j, i)$ be the edge of T coming out of j. Then $i <_{\Pi(T)} j$, so i is not an element of L by choice of j. For each of the a_j edges $e = (j, i')$ such that $e <_{\Pi(T)} e_j$, we have $i' \leq_{\Pi(T)} i <_{\Pi(T)} j$, so i' is not an element of L. Thus there are at least $a_j + 1$ edges going from j to vertices outside of L.*

Next, we define the inverse map $g_{\Pi,G} : P_G \longrightarrow S_G$. Given $P = (a_1, \ldots, a_n) \in P_G$, we construct the corresponding tree $g_{\Pi,G}(P)$ one edge at a time. Initially, let t_0 be the subtree of G consisting of the vertex 0 alone, and put $p_0 = 0$. For $1 \leq k \leq n$, We choose the vertex p_k and construct the subtree t_k rooted at 0 inductively as follows. Let L_k be the set of vertices not in t_{k-1}, and let W_k be the set of vertices $j \in L_k$ such that the number of edges from j to t_{k-1} is at least $a_j + 1$. Note that $|W_k| \geq 1$ by definition of a G-parking function. For each $j \in W_k$, let e_j be the from j to t_{k-1} such that exactly a_j edges e from j to t_{k-1} satisfy $e <_{\Pi(t_k - 1)} e_j$. Let t be the tree obtained by adjoining each vertex $j \in W_k$ to t_{k-1} by means of the edge e_j. Set P_k to be the smallest vertex of W_k in the order $\Pi(t)$, and set t_k to be the tree obtained by adjoining p_k to t_{k-1} by means of the edge e_{pk}. Obviously, t_k is a subtree of G. In the end set $g_{\Pi,G}(P) = T = t_n$. For the rest section, we write g instead $g_{\Pi,k}$.

Example 20. *Let G be the graph show in the figure, and let p=(0,1,0,1). Let Π be the tree order in which vertex $i < j$ if i is closer to the root than j, or else if i and j are equidistant to the root, and $i < j$. Initially, $L_1 = \{1, 2, 3, 4\}$ and $W_1 = \{1\}$, so vertex 1 is attached to the root to produce the subtree t_1. Then we have $W_2 = \{3, 4\}$ with $e_3 = (3, 1)$ and $e_4 = (4, 1)$. Adjoining vertices 3 and 4 to t_1 by means of e_3 and e_4 places vertices 3 and 4 the same distance away from the root, making $p_2 = 3$, so vertex 3 attached by means of the edge $e_3 = (3, 1)$ to produce t_2. At the next step, we have $W_3 = \{2, 4\}$ with $e_2 = \{2, 3\}$ and $e_4 = (4, 1)$. Adjoining vertices 2 and 4 to t_2 by means of e_2 and e_4 makes vertex 4 closer to the root than vertex 2, so we select vertex 4 and attach it to vertex 1 to form t_3. Finally, we attach vertex 2 to 3 to form $t_4 = g(p)$.*

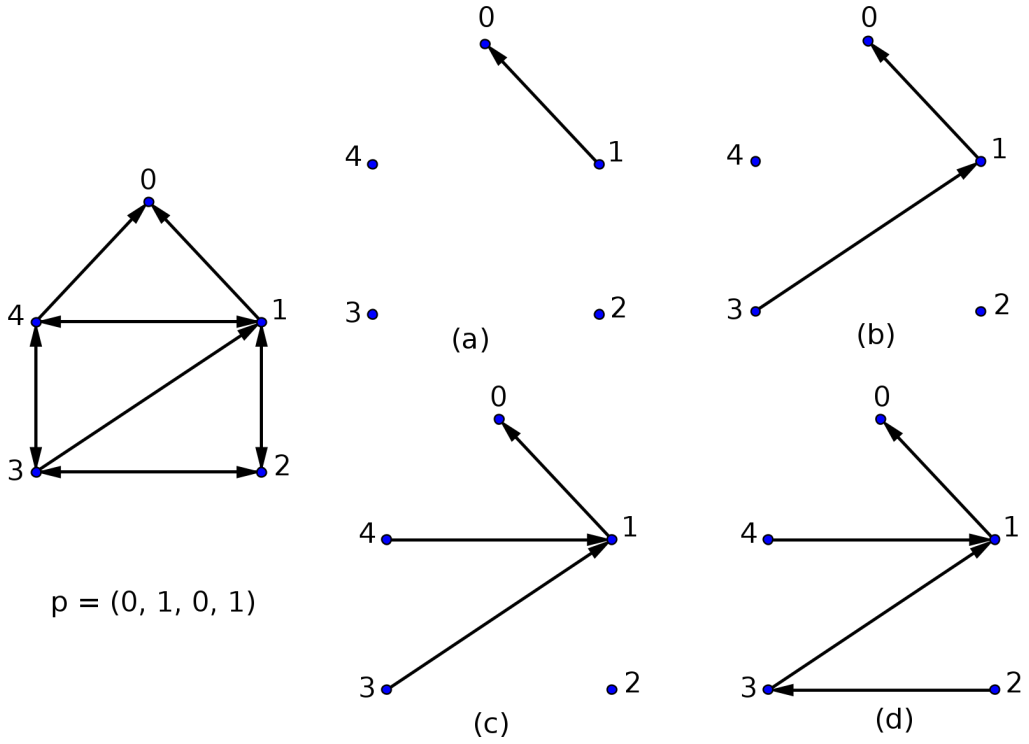


Figure 4.8: Directed graph

Lemma 5. *In the above construction, $p_0 <_{\Pi(T)} \ldots <_{\Pi(T)} p_n$.*

proof 15. *Since $\Pi(G)$ is a proper set of tree orders, it follows that the root 0 is the smallest vertex of T in the order $\Pi(T)$. Hence $p_0 <_{\Pi(T)} p_1$,. Suppose that $p_0 <_{\Pi(T)} \ldots <_{\Pi(T)} p_k$ for some $1 \leq k \leq n-1$. We show that $p_k <_{\Pi(T)} p_{k+1}$. We consider the following two cases.*
Case 1: p_{k+1} is not an element of W_k. Then the number of edges from p_{k+1} to t_{k-1} is at most b_{p_k+1}. Since $p_{K+1} \in W_{k+1}$, the number of edges from p_{k+1} to t_k is at least $a_{p_{k+1}} + 1$. It follows that there is at least one edge (p_{k+1}, p_k) in G and p_{k+1} is adjoined to t_k by means of such an edge. Thus $p_k <_{\Pi(T)} p_{k+1}$ because $\Pi(G)$ is a proper set of tree orders.
Case 2:$p_{k+1} \in W_k$. Let e_{pk+1} be the edge from p_{k+1} to t_{k-1} such that exactly b_{Pk+1} edges e from P_{k+1} to t_{k-1} satisfy $e <_{\Pi(t_k-1)} e_{pk+1}$. Since P_k is the largest vertex of t_k in the order $\Pi(T)$ and hence in the order $\Pi(t_k)$ and e_{p_K+1} goes from p_{k+1} to $t_{k-1} = t_k - p_k$, it follows that

$e <_{\Pi(t_k-1)} e_{p_k+1}$ *if and only if* $e <_{\Pi(t_k)} e_{p_k+1}$ *because the order* $\Pi(t_k-1)$ *is consistent with* $\Pi(t_m)$. *Therefore, exactly* a_{p_k+1} *edges* e *from* p_{k+1} *to* t_k *satisfy* $e <_{\Pi(t_k)} e_{p_k+1}$, *hence* p_{k+1} *is adjoined to* t_k *by means of the edge* e_{p_k+1}.
Let e_{pk} *be the edge of* T *coming out of* p_k, *and let* t *be the tree in the construction of* T *obtained by adjoining the vertices of* W_k *to* t_{k-1}. *Let t' be the tree obtained from* t_{k-1} *by adjoining the vertices* p_k *and* p_{k+1} *by means of the edges* e_{pk} *and* e_{pk+1}. *Then t' is a subtree of both t and T. By choice of* p_k, *we have* $p_m <_{\Pi(t)} p_{m+1}$, *so* $p_m <_{\Pi(t')} p_{m+1}$ *and* $p_m <_{\Pi(t)} p_{m+1}$ *because the order* $\Pi(t')$ *is consistent with both* $\Pi(t)$ *and* $\Pi(T)$.
Next we check that f and g are inverses each other.

Lemma 6. *f(g(p))=p for* $p \in P_G$.

proof 16. *Put* $P = (a_1, \ldots, a_n)$ *and* $T = g(P)$. *Consider the process of constructing T. For* $j \in \{1, \ldots, n\}$, *we have* $j = p_k$ *for some* $1 \leq k \leq n$. *Let* e_j *be the edge of T coming out of j. The edge* e_j *goes from j to* t_{k-1}. *Since the set of vertices of* t_{k-1} *is* $\{p_0, \ldots, p_{k-1}\}$, *it follows from lemma 4 that if an edges e coming out of j satisfies* $e <_{\Pi(T)} e_j$, *then e goes from j to* t_{k-1}. *Thus,* $e <_{\Pi(T)} e_j$ *if and only if* $e <_{\Pi(t_{m-1})} e_j$ *because the order* $\Pi(t_{k-1})$ *is consistent with* $\Pi(T)$. *By construction of T, the number of edges e satisfying* $e <_{\Pi(t_{k-1})} e_j$ *is* a_j, *hence the number of edges e satisfying* $e <_{\Pi(T)} e_j$ *is also* a_j. *We conclude that f(T)=P.*

Lemma 7. $g(f(T')) = T'$ *for* $T' \in S_G$

proof 17. *Put* $P = f(T') = (a_1, \ldots, a_n)$. *Consider the process of constructing* $T = g(P)$. *We show by induction that for* $0 \leq k \leq n$,*the tree* t_k *is a subtree of T' and that* $p_0, \ldots, p_k$ *are the smallest k+1 vertices in the order* $\Pi(T')$. *Since the root 0 is the smallest vertex in* $\Pi(T')$, *the assertion is true for k=0.*
Now, suppose that t_{k-1} *is a subtree of T' and that* $p_0, \ldots, p_{k-1}$ *are the smallest k vertices in the order* $\Pi(T')$. *Let m be the* $(k+1)^{th}$ *smallest vertex in the order* $\Pi(T')$, *and let* $e'_m = (m, i)$ *be the edge coming out of m in T'. Then* $i <_{\Pi(T')} m$, *so* $i \in \{p_0, \ldots, p_{k-1}\}$ *and* $i \in t_{k-1}$. *Hence if an edges e coming out of m satisfies* $e <_{\Pi} (T')e'_m$, *then e goes from m to* t_{k-1}. *There are* a_m *edges e satisfying* $e <_{\Pi} (T')e'_m$. *These* a_m *edges together with the edge* e'_m *give* $a_m + 1$ *edges going from m to* t_{k-1}. *It follows that* $m \in W_k$.
As before, for every $j \in W_k$, *let* e_j *be the edge from j to* t_{k-1} *such that exactly* a_j *edges e from j to* t_{k-1} *satisfy* $e <_{\Pi(t_{k-1})} e_j$. *Since the vertices of* t_{k-1} *are the smallest k vertices in the order* $\Pi(T')$, *it follows that if an edge e coming out of j satisfies* $e <_{\Pi(T')} e_j$, *then e goes from j to* t_{k-1}. *Thus,* $e <_{\Pi(T')} e_j$ *if and only if* $e <_{\Pi(t_{k-1})} e_j$ *because the order* $\Pi(t_k-1)$ *is consistent with* $\Pi(T')$. *There are* a_j *edges satisfying* $e <_{\Pi(t_{k-1})} e_j$, *hence there are* a_j *edges e satisfying* $e <_{\Pi(T')} e_j$. *It follows from the choice of* a_j *that* e_j *is an edge of T'. Therefore, the tree i obtained by adjoining the vertex* $j \in W_k$ *by means of the edges* e_j *is a subtree of T'. Consequently, the smallest vertex* p_k *of* W_k *in the order* $\Pi(t)$ *is the smallest vertex of* W_k *in the order* $\Pi(T')$. *Since m is the smallest vertex of* L_k *in the order* $\Pi(T')$ *and* $m \in W_k \subseteq L_k$, *it follows that* $p_k = m$. *The induction step is complete.*
Finally, we obtain $T' = t_n = T$.
Theorem 12 follows from the last two lemmas.

Example 21. *Let G be a directed graph*
$(0,0,0), (1,0,0), (0,1,0), (0,0,1), (1,1,0), (0,1,1)$ *are* $G-$*parking functions of the given digraph. And also we have 6 different spanning trees.*

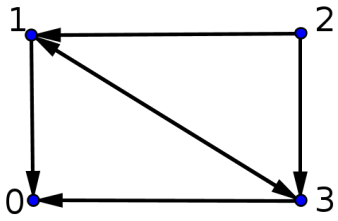


Figure 4.9: Directed graph

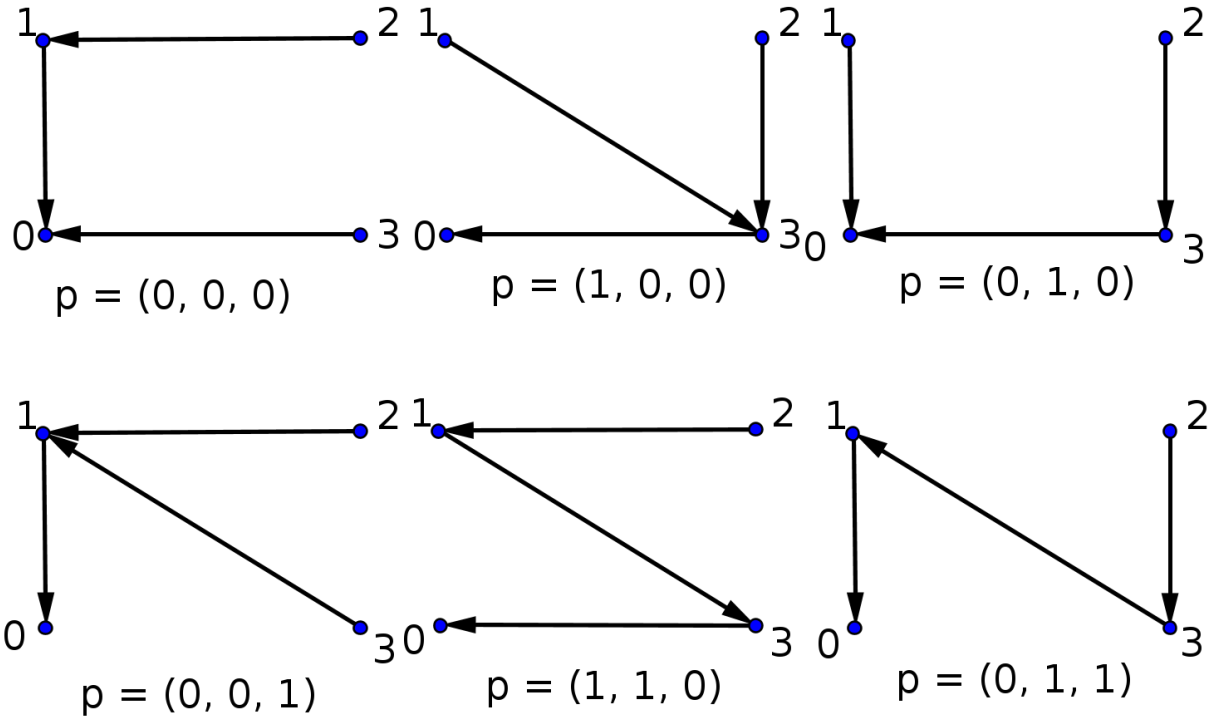


Figure 4.10: The corresponding spanning tree for the given directed graph.

Theorem 13. *For each parking function $a = (a_1, a_2, \ldots, a_n)$, define the difference sequence $(c_1, c_2, \ldots, c_{n-1})$ by letting*

$$c_i = a_{i+1} - a_i (mod n + 1) \tag{4.1}$$

Let T(a) be the labeled tree in T_{n+1} whose prüfer code is $(c_1, c_2, \ldots, c_{n-1})$. Then the map $a \longrightarrow T(a)$ is a bijection between set of parking function and T_{n+1}.
$PK_n \equiv$ set of parking functions of length n a bijection from PK_n to $([n]_0)^{n-1}$

proof 18. *The inverse equation of 4.1 can be written as;*

$$a_i = a_1 + c_1 + \ldots + c_{i-1} (mod n + 1) \tag{4.2}$$

$2 \leq i \leq n$.

That is, a parking function is determined by its difference sequence if a_1 is known. The following algorithm describes now to find an a_1 for each $(c_1, c_2, \ldots, c_{n-1}) \in \{0, 1, \ldots n\}^{n-1}$ such that $(a_1, a_2, \ldots, a_n)$ determined by equation 4.2 is a parking function.

The Classical parking functions are defined in the following ways. There are n drivers, labeled $\{1, 2, \ldots, n\}$ and n parking spots $\{0, 1, 2, \ldots, n-1\}$ arranged linearly in this order. Each driver i has a favorite parking spot b_i. Drivers enter the parking area in the order in which they are labeled. Each driver proceeds to his favorite spot and parks there if it is free, or parks at the next available spot otherwise. The sequence $\{b_1, b_2, b_3, \ldots, b_n\}$ is called a parking functions if every driver parks successfully by this rule. The most notable result about parking functions is a bijective correspondence between such functions and tree on $n+1$ labeled vertices.

A spanning tree rooted at m is a subgraph of G such that, for each $i \in \{0, 1, 2, \ldots, n\}$, there is a unique path from i to m along the edges of the spanning tree. Note that these are the spanning trees of the graph in the usual sense with each oriented towards m. The number of such trees is given by the matrix-tree theorem; see[19]. In [15], it is shown that the number of spanning trees of G rooted at 0 is equal to the number of G-parking functions for any digraph G.

An equivalent fact was originally discovered by Dhar[5], who studied the sandpile model. the so-called recurrent states of the sandpile model are in one-to-one correspondence with G-parking functions for certain graphs G, including all symmetric graphs G is mentioned in [10], and a class of bijection is constructed in [4]. The sandpile model was also studied by Gabrielov in [8].

There is another generalization of classical parking function that goes under the unfortunately vague name of generalized parking functions where as graphical parking functions depend on a choice of graph G, generalized parking functions depend on a choice of vector $X = (x_1, \ldots, x_n)$ of non-negative integers. As such we will call them parking functions. Let $X = (x_1, \ldots, x_n)$ a sequence of positive integers. An X-parking function is a sequence $(a_1, \ldots, a_n)$ of positive integers whose non-decreasing rearrangement $b_1 \leq b_2 \leq \ldots \leq b_n$ satisfies $b_i \leq x_1 + \ldots + x_i$. We denote the set of X-parking function by PF(X). Vector parking functions have been studied in many subsequent papers, especially by Yan and her collaborators. Ordinary parking functions are the case $X = (1, 1, 1, \ldots, 1)$.

4.3 Parking Functions and Rooted Labelled Trees

Definition 4.6. *Let T be a rooted tree with vertices labeled $\{1, 2, 3, \ldots, n\}$. Let $C_1, \ldots, C_n$ be the n cars and assume that C_i prefers the vertex with label u_i. The sequence $(u_1, \ldots, u_n)$ is a T-parking function if and only if all the cars can find a parking space in T.*

The number of parking function of length n and the number of labeled trees on the vertex set $\{0, 1, \ldots, n\}$ is $(n+1)^{n-1}$.

Remark 1. *We denote a rooted tree by (T, r), where r is the root of T.*

Example 22. *Suppose we have a size 8 rooted labeled tree T and a sequence u=(2,8,7,1) of addresses of preferred parking space for 4 drivers. All drivers are successful, thus (T, u) yields a $(8, 4)$ tree parking function; the parking positions of the drivers are give by the sequence $(2, 8, 7, 1)$.*

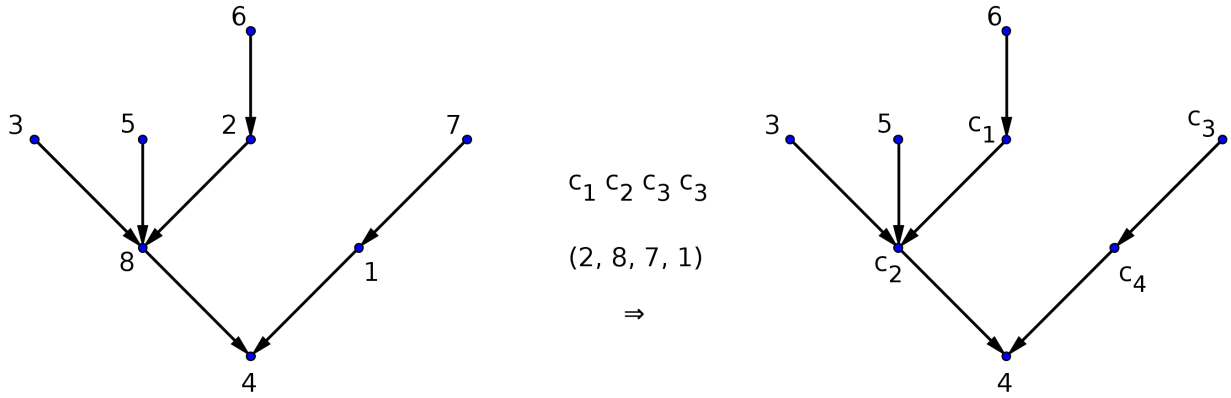


Figure 4.11: Rooted labeled tree.

Conversely, the sequence $(5, 8, 4, 4)$ is not a parking function for T, since the fourth driver is not able to park.

Example 23. *Suppose we have a size 6 rooted labeled tree T and a sequence $u = (2, 2, 1, 4, 2, 5)$. The sequence u is a parking function since all drivers are able to park. We can see figure 4.12*

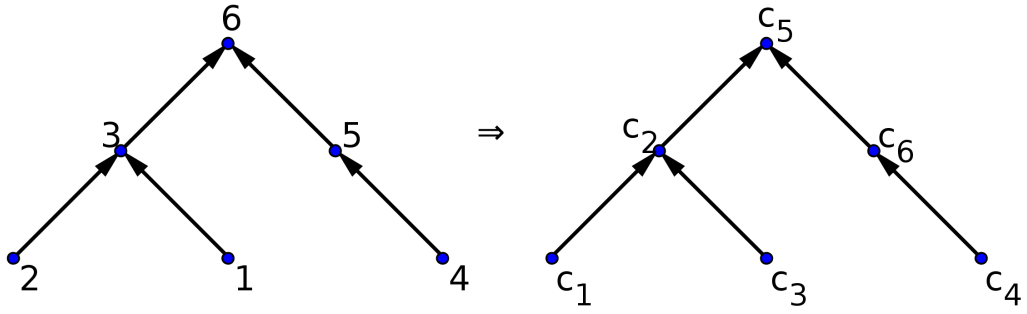


Figure 4.12: Rooted labelled tree

Definition 4.7. *For a parking function (T, r), we say that an edge e is used by r if there exists some driver who, after failing to park at her preferred spot, crosses e during her search for an unoccupied spot.*

The following figure shows a classical parking function with both used and unused edges. Recall that a characterization of classical parking functions of length n is that there are at least i items less than or equal to i in the preference sequence. For tree parking functions have a similar characterization.

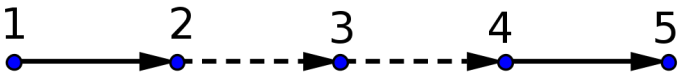


Figure 4.13: Classical parking function.

4.4 Parking Functions and Complete Bipartite Graphs

Let $N = \{0, 1, 2, 3, \ldots\}$. Recall that a parking function of length n is a sequence $(a_1, a_2, \ldots, a_n) \in N^n$ whose weakly increasing rearrangement $b_1 \leq b_2 \leq \ldots \leq b_n$ satisfies $b_j \leq j - 1$ for all $1 \leq j \leq n$. Let PF(n) denote the set parking functions of length n. Let K_n denote the complete graph of n vertices. We have seen that well established result which relates number of parking function of length n and the number of spanning trees of K_{n+1}. That is

The number of PF(n) = The number of spanning trees of $K_{n+1} = (n+1)^{n-1}$ .....(1)

Furthermore, there are many explicit bijections between parking function and labeled trees; with variant properties which are very useful.
Question:(1) What is a bipartite analog of this equation ?
Claim: The number of even parking functions of $PF(2m-1)$ is equal to the number of spanning trees of $K_{m,m}$ is equal to m^{2m-2}, where $K_{m,m}$ denote the complete bipartite graph with m vertices in one part and m vertices in another part.

That is; The number of $\{(a_1, a_2, \ldots, a_{2m-1}) \in PF(2m-1) : a_i$ is even for all i $\}$ = The number of spanning trees of $K_{m,m}$= m^{2m-2}

Example 24. *1,4,81,4096,...*

proof 19. *Let $n = 2m$ and let $(a_1, a_2, \ldots, a_{2m-1})$ be an even parking function. Using the previous Pollak's map:*
C=$(a_2 - a_1, \ldots, a_{2m-1} - a_{2m-2})$(mod 2m). Thus, C is an (2m-2)-tuple of even numbers between and 2m-1.
Let $b = (\frac{c_1}{2}, \ldots, \frac{c_{m-1}}{2})$ and

$d = (\frac{c_m}{2} + m, \ldots, \frac{c_{2m-2}}{2} + m)$.

Then (b,d) is the bipartite prüfer code of a spanning tree of $K_{m,m}$, where the vertices of one block are labelled $0, \ldots, m-1$ and the vertices of the other block are labelled $m, \ldots, 2m-1$.

The Bijection

Consider the bijection between factorizations of the long cycle $(1, \ldots, 2m)$ and parking functions of length 2m-1. Restrictions made to a bijection:
First: Let $1 \leq a_i \leq j$ rather than $0 \leq a_i < j$ and consider the parking functions of length 2m-1 with all parts being odd.

Lemma 8. *A parking function $P = b_1 b_2 \dots b_{2m-1}$ has all parts odd if and only if the factorization. $\phi^{-1}(P) = (i_1 j_1) \dots (i_{2m-1} j_{2m-1})$ has the property that all first entries $i_1, \dots, i_{2m-1}$ are odd .*
It follows, that the bijection ϕ restricts to a bijection between factorization of the long cycle $(1, \dots, 2_m)$ into transpositions $(i_k j_k)$ with all i_k odd and parking functions of length $2m-1$ with all parts being odd.

Open problems:

1. Using the above Lemma and observation how to find a bijection between such factorization and spanning trees of $K_{m,m}$.

2. What is $K_{n,m}$, $n \neq m$?

Note: Factorization of the long cycle $(1, 2, \dots, n) \in S_n$ into a product of n transpositions (ij). The map from these factorizations to parking functions m is the map ϕ sending $C = (i_1 j_1) \dots (i_n j_n)$ to the sequence $(i_1, \dots, i_n)$. In Stanley's article of 1996, it was shown that $\phi(factorization)$ is indeed a parking function, and that this map is a bijection. To obtain $\psi := \phi^{-1}$ we must therefore start with a sequence $(i_1, \dots, i_n)$ and turn it into a factorization $C = (i_1 j_1) \dots (i_n j_n)$.
This can be done in two steps:

First, define ψ for primitive parking functions (weakly increasing), and then obtain ψ for general parking functions by a sequence of transpositions of factors

$(i_k j_k)(i_{k+1} j_{k+1}) \longmapsto (i_{k+1} j_{k+1})(i_k \overline{j_k})$, such that $(i_k j_k)(i_{k+1} j_{k+1}) = (i_{k+1} j_{k+1})(i_k j_k)$
It would be observed that this uniquely determines $\overline{j_k}$, and that this is not a valid procedure for any two transpositions. On the other hand, this works for sequences coming from parking functions.

Example 25. *Let $seq = (i_1, \dots, i_n)$ be a primitive parking function $\psi(seq)$ is then given by sending the last 1 in seq to the transposition (12). Afterwards, send the last $i \in \{1, 2\}$ to (i,3), then the last $i \in \{1, 2, 3\}$ to (i,4) and so on. For instance, replacing letters in 112355 in this way gives.*
$112355 \longmapsto 1(12)2355 \longmapsto 1(12)(23)355 \longmapsto 1(12)(23)(34)55 \longmapsto (15)(12)(23)(34)55 \longmapsto (15)(12)(23)(34)5(56) \longmapsto (15)(12)(23)(34)(57)(56)$
Now, to obtain the parking function 211553, we first interchange positions 2 and 3, and interchange the factors such that $(i_2 j_2)(i_3 j_3) = (i_3 j_3)(i_2 \overline{j_2})$, so we get

$\psi(121355) = (15)(23)(13)(34)(57)(56)$
Here, the 3^{rd} factor became the 2^{nd}, and the 2^{nd} factor (12) turn into (13). Keep going with this:
$\psi(112355) = (15)(12)(23)(34)(57)(56)$
$\psi(121355) = (15)(23)(13)(34)(57)(56)$
$\psi(211355) = (23)(15)(13)(34)(57)(56)$
$\psi(213155) = (23)(15)(34)(14)(57)(56)$
$\psi(213515) = (23)(15)(34)(57)(14)(56)$

$\psi(213551) = (23)(15)(34)(57)(56)(14)$
This shows the construction and the procedure work in both steps.

It would be reasonable to conjuncture that the number of spanning trees of $K_{m,n}$ is $n^{m-1}m^{n-1}$ based on the above discussion takes m=n. Where as the even parking functions described are (after dividing by two). (2m-1,m)-rational parking functions; it is known that the number of (a,b)-parking functions with gcd(a,b)=1 is b^{a-1}.

Chapter 5

Conclusion

We have discussed some bijections of parking functions and some combinatorial objects like, maximal chain of non-crossing partitions, rooted labelled trees, G-parking functions and also we have established a new relationships between parking functions and spanning trees of complete bipartite graph. We also have reviewed some preliminaries, including graph and graph invariants.

We have shown the existence of basic techniques that shows the number of tree on $[n+1]$ vertices and the number of parking function of length n are the same. We came across the existence of many explicit bijections between parking function and labeled trees; with variant properties which are very useful. However, question like: "What is a bipartite analog of number of parking function of length n and the number spanning trees of K_{n+1} ?" was claimed as "The number of even parking functions of $PF(2m-1)$ is equal to the number of spanning trees of $K_{m,m}$ is equal to m^{2m-2}, where $K_{m,m}$ denote the complete bipartite graph with m vertices in one part and m in another part." and have been tried to justify the claim.

There are some open questions about parking functions that would be interesting to look into. Among them: is there any bijection between parking functions and bipartite graphs ? Is there any alternative explicit proof for the number of spanning trees of $K_{m,n}$ that was conjectured to be $n^{m-1}m^{n-1}$?

However, there are certainly more aspects in the subject that are not discussed here. Readers interested to learn more are invited to read the references for a very extensive treatment of parking functions.

Acknowledgments

First and foremost, praises and thanks to the Allah, the Almighty, for his provision and protection in all aspects of my life. Next I would like to express my special thanks of gratitude to AAU (Addis Ababa University) and ISP (International Science Program) for their academical and financial support.

I would like to express my special thanks to my supervisor Dr. Samuel Assefa. I appreciate his contributions of time and idea to make my work productive and stimulating. I am amazed that even during his busy schedule, he will never say no, instead he was always willing to help me with my thesis.

I am extremely grateful to my parents for their love, duas, caring and sacrifices for educating and preparing me for my future. Words can't express the feeling I have for my mom(Matey) and my brother (Awet) for their constant unconditional support in my study. Matey you have been a constant source of strength and inspiration to me. Thanks mom, your Dua for me was what sustained me this far and put me through from many difficulties that I am facing for all these years.

I also thank all the departmental staff for helping me during these study both academically and officially. I would like to say thanks to my friends and research colleagues, special thanks to Fufa for all your help and support.

Finally, my thanks go to all the people who have supported me to complete the research work directly and indirectly.

Bibliography

[1] H.W. Becker, (1952). Planar rhyme schemes. *Bell.Amer.Math.* soc.**58**:39.

[2] J.S. Beissinger and U.N. peled, (1996). A note on major sequences and external activity in trees. *The Electronic Journal Of Combinatorics* **3**.

[3] P.Biane,(2002). Parking functions of types a and b. *The Electronic Journal Of Combinatorics* .**9**.

[4] R. Cori, Y. Le Borge, (2003). The sand-pile model and tutte polynomials. *Add.Appl. Math.***30**:44-52.

[5] D.Dhar, (1990). Self-organized critical state of the sand plle automaton models.*Phys.Rev.Lett.***64**:1613-1616.

[6] P.H. Edelman, (1980). Chain enumarative and non-crossing partitions. *Discrete Mathematics*.171-180.

[7] J.Francon, (1975). A cyclic and parking functions. *J. Combin. Theory Ser. A* **18**:27-35.

[8] A. Gabrielov, (1993). Asymmetric abelian avalanches and sandpile. preprint.MSI, *cornell university*.93-65.

[9] J.D. Gilbey and L.H. Kalikow, (1999). Parking functions. Valet functions and priority queus. *Discrete mathematics* **197-198**: 351-373.

[10] E.V. Ivashkevich and V.B. Priezzer, (1998). Introduction to the sandpile model. *physica A*. **254**:97-116.

[11] A.G. Konheim and B Weiss, (1966). An occupancy dicipline discipline and applications. *SIAM J.Appl. Math***14**:1266-1274.

[12] G.Kreweras,(1972). Surles partitions non-crossig d'un cycle. *Descrete Math.***1**:333-350.

[13] J. Kung and C.H. Yan,(2003). Exact formula for moments of sums of classical parking functions. *Advances in Applied Mathematics* **31**:215-241.

[14] M.L.Lackner and A.Panholzer, (2016). parking functions for mappings. *J.combin. Theory Ser.A* **142**:1-28.

[15] A. Postnikov and B. Shapiro, (2004). Trees, parking functions, Syzgies, and deformations of monomial ideal. *Trans.Amer. Math. Soc.* **356**:3109-3142.

[16] Y. Poupard,(1972). Etude et denombrement parallel des partitions non crossing d'un cycle et des decoupage d'un polgone convexe. *Discrete Math.* **2** : 279-288.

[17] J.Riordan,(1969). Ballots and trees. *J. comb Theory.***6** : 408-411.

[18] R.Simion and D.Ullman, (1991). On the structure of the lattice of non-crossing partitions. *Discrete Mathemtics*. **98**:193-206.

[19] R.P. Stanley, (1999).*Enumerative Combinatorics*. Cambridge University Press. Cambridge. Vol (**2**).

[20] C.H. Yan and D.Kostic, (2008). Multiparking functions, graph search, and tutte polynomial.*Advances in Applied Matematics* .**40**: 73-97.

[21] C.H.Yan,(1997). Generalized tree inversions and K-parking functions. *Journal of combinatory theory* **79**:268-280.